Title

COMPUTER CONTROL OF ELECTROCHEMICAL EXPERIMENTS WITH APPLICATION TO ZINC/NICKEL OXIDE CELLS

Authors

McLarnon, F.R.
Cairns, E.J.

Publication Date

1982-12-01

LBL-15546

Lawrence Berkeley Laboratory

UNIVERSITY OF CALIFORNIA

ENERGY & ENVIRONMENT DIVISION

COMPUTER CONTROL OF ELECTROCHEMICAL EXPERIMENTS WITH APPLICATION TO ZINC/NICKEL OXIDE CELLS

Marco H. Katz*, Frank R. McLarnon,
and Elton J. Cairns

(*M.S. Thesis)

December 1982

ENERGY AND ENVIRONMENT DIVISION

Prepared for the U.S. Department of Energy under Contract DE-AC03-76SF00098

DISCLAIMER

LBL-15546

COMPUTER CONTROL OF ELECTROCHEMICAL EXPERIMENTS WITH APPLICATION TO ZINC/NICKEL OXIDE CELLS

by

Marco H. Katz, Frank R. McLarnon, and Elton J. Cairns

December 1982

Energy and Environment Division
Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

This work was supported by the Assistant Secretary for Conservation and Renewable Energy, Office of Energy Systems Research, Energy Storage Division of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

TABLE OF CONTENTS

Computer Control of Electrochemical Experiments with Application to Zinc/Nickel Oxide Cells

Marc H. Katz

University of California

Berkeley, CA 94720

ABSTRACT

A computer-controlled test system has been designed and constructed to allow the simultaneous and continuous cycling of 16 or more electrochemical cells. The system offers resolution and stability within 0.025% of full-scale, and response times are typically 1 ms. A wide variety of charge and discharge regimes, including pulsed charge and power discharge profiles, are implemented through various computer algorithms. The software configuration allows two programs to execute concurrently in the computer memory and to communicate with each other through a set of codified messages. This arrangement permits flexible interaction with all experiments and provides on-line data reduction and display.

The test system has been employed to investigate the effects of constant-current and various pulsed-current charging modes on the cycle life and capacity retention of Zn/NiOOH cells cycled at 100% depth-of-discharge. Test results included cell voltages, potentials of the NiOOH and Zn electrodes versus Hg/HgO reference electrodes, coulombs, watt-hours, average voltages, and associated efficiencies. Post-mortem tests included Zn electrode X-ray studies.

A 30ms-ON/90ms-OFF pulse profile, with a 15mA/cm^2 peak current density, improved the operating conditions of the cells by minimizing the overpotentials on the Zn and NiOOH electrodes. In contrast to all other charge profiles tested, it eliminated cell shorting (for at least 100 cycles) and slowed the loss of ZnO reserve, thereby preventing rapid cell capacity decline.

CHAPTER 1

INTRODUCTION

The purpose of this work was to design, build, and implement a minicomputer-based test system to control and monitor a variety of electrochemical experiments. An integrated hardware and software design led to the construction of a multiple-channel electrochemical cell test system, which was adapted for the cycle-life testing of experimental tri-electrode cells arranged in a configuration similar to that expected in secondary batteries. Experiments were conducted to determine the effect of various charging profiles on the lifetime of Zn/NiOOH cells.

1.1 THE Zn/KOH/NiOOH SYSTEM

1.1.1 Overall Reactions

The overall chemical reaction for the Zn/KOH/NiOOH system is (1)

$$2NiOOH + Zn + H_2O \underset{C}{\overset{D}{\rightleftharpoons}} 2Ni(OH)_2 + ZnO$$

The electrochemical half-reaction at the positive electrode is

$$2NiOOH + 2H_2O + 2e^- \underset{C}{\overset{D}{\rightleftharpoons}} 2Ni(OH)_2 + 2OH^-$$

The electrochemical half-reaction at the negative electrode is

$$Zn + 2OH^- \underset{C}{\overset{D}{\rightleftharpoons}} ZnO + H_2O + 2e^-$$

The theoretical cell voltage is 1.74 V and the theoretical specific energy is 345 Wh/kg, calculated on the basis of the Gibbs free energy change and the mass of the reactants.

1.1.2 APPLICATIONS

The Zn/NiOOH battery is one of the candidate systems under consideration for the Near-Term Electric Vehicle Battery Project, sponsored by the U. S. Department of Energy. Battery performance and cost objectives (specified for electric vehicle applications) are compared with experimental results obtained prior to 1981 (2) in Table 1. Other potential commercial applications for the Zn/NiOOH system include automotive SLI batteries and other uses where this battery may replace Pb/PbO_2 or Cd/NiOOH batteries.

1.2 CHARACTERISTICS OF THE Zn/NiOOH SYSTEM

The Zn/NiOOH cell offers attractive specific energy and specific power characteristics: its shortcomings are high cost and unsatisfactory cycle life (3). These weaknesses result from particular characteristics of both the NiOOH and the Zn electrodes.

1.2.1 THE NICKEL OXIDE ELECTRODE

At present, only NiOOH electrodes based on a sintered Ni plaque have shown satisfactory performance and cycle life (3) when tested in Zn/NiOOH cells, although attempts are being made to reduce the cost of this battery by using plastic-bonded NiOOH electrodes (2). The NiOOH electrode operates at less than full efficiency due to the occurrence of a parasitic reaction (oxygen evolution) when the electrode is charged.

Slow evolution of oxygen also causes self-discharge of the NiOOH electrode (1).

1.2.2 THE ZINC ELECTRODE

1.2.2.1 An overall reaction proposed for the anodic process at the Zn electrode follows a dissolution-precipitation mechanism (4).

$$Zn + 4OH^- \rightarrow Zn(OH)_4^{2-} + 2e^- \quad \text{Dissolution}$$

$$Zn(OH)_4^{2-} \rightarrow ZnO + 2OH^- + H_2O \quad \text{Precipitation}$$

The dissolution reaction consumes four OH^- ions, while the NiOOH electrode discharge reaction produces only two. Thus, a critical factor during discharge is the presence of a quantity of electrolyte in the pores of the Zn electrode sufficient to maintain the dissolution reaction rate at an acceptably high level (5).

1.2.2.2 A generic problem of the Zn electrode is the high solubility of its discharge products in KOH. High concentration of the Zn species in the electrolyte is linked to the redistribution of Zn upon cycling (6). During discharge, a 31% KOH electrolyte can reach more than three times the ZnO saturation concentration (7).

1.2.2.3 The Zn electrode can also evolve hydrogen on charge, but the addition of metals such as mercury, lead, cadmium, or thallium has been found to suppress (6) the undesirable side reaction.

1.2.3 FAILURE MECHANISMS

The major failure mechanisms of a Zn/NiOOH cell are associated with its Zn electrode (3). The most important are: shape change (8,9), densification (6), passivation (10), and dendrite shorting (6).

Several operational parameters that influence the rate of shape change have been described elsewhere (6). Various attempts have been made to reduce the Zn electrode solubility in KOH electrolyte, e.g., additives to the electrode (11) or additives to the electrolyte (12). The rate of dendrite shorting has been reduced by various methods such as the operation of sealed cells (13) and the development of separators that more effectively slow the rate of dendrite penetration (14,15).

1.3 OPERATING CONDITIONS

1.3.1 DISCHARGE MODES

An electric vehicle battery is typically subjected to a variable load, due to conditions of vehicle acceleration or deceleration, steep road grades, etc. A realistic test of an experimental cell designed for this application is a controlled discharge that employs a power profile, equivalent to a standard driving profile, adapted to the capacity of the cell. A typical example is the EPA urban driving profile (16) which consists of a power load which varies every second, over a total of 1372 seconds. The implementation of such a test is well within the capability of modern computer systems. A description of this application can be found in Appendix G.

Simplified driving profiles have been proposed which retain some realistic testing conditions and make the implementation more straightforward (17). However, most of the battery tests are still conducted at constant current or constant power, conditions which are more appropriate for load-leveling applications.

1.3.2 CHARGE MODES

Contrary to discharge modes, there is no basic limitation on the variety of possible charge modes for a battery. Important practical considerations are the availbility of hardware and sufficient recharge time.

> Constant-current charge is popular because of the straightforward implementation (1). In the Zn/NiOOH cell, when the end of a charge is approached, gas evolution rates increase, overpotential on the Zn electrode rises, and an unwelcome dendrite growth condition is established.

> During a constant-voltage charge, the current gradually decreases with increasing time, which prevents excessive overpotential on the electrodes. The resulting low currents, however, lead to longer recharge times. For that reason, a combination of constant-current and constant-voltage is often used (constant voltage-limited current charge).

> Other methods have been proposed, such as stepped constant-current, floating, trickle (1), and pulsed-current charging. This last method is discussed in the next section.

1.4 PULSED CURRENT CHARGING OF Zn/NiOOH CELLS

Although pulsed current has not until recently been systematically evaluated as an alternate charge method for Zn/NiOOH cells, it has long been known to improve the characteristics of electroplated deposits of various metals (18).

Romanov (19) described how the growth of Zn dendrites in alkaline Zn/NiOOH and Zn/AgO batteries could be decreased by using AC current superimposed on DC current. His observations indicated that the best results were obtained for pulsed currents having a sufficiently long off-time, or sufficiently low frequency (<250Hz). Bennion (20) suggested that improvements in dendrite growth suppression with pulsed-current charging might be due to a combination of a short, high-current peak, followed by a sufficiently long rest-time. A high-current peak creates a relatively large surface overpotential, which increases the number of nucleation sites and thereby reduces the chance for dendrite initiation. The rest-time allows local zinc concentration gradients to relax by diffusion of zinc ions into the depleted diffusion layer, which prevents the development of excessive concentration polarization, hydrogen evolution and dendrite growth.

Chin et. al. (21) studied the effects of pulsed current on the deposition of Zn from an acid zinc chloride solution. They found that pulsed currents produced an increase in nucleation sites and therefore more uniform deposits. The frequency employed for these experiments was 60Hz.

Smithrick (22) described experiments with various pulsed-current charge profiles and their effect on the cycle life of Zn/NiOOH cells. He was unable to demonstrate that any pulsed charge method improved cell cycle life over that associated with constant current. These tests were conducted at frequencies of 120 to 1000 Hz, and the short off-times of these pulses might explain the lack of lifetime improvement.

By contrast, O. Wagner (14) presented results which showed that pulsed-current charge could, indeed, improve the cycle life of Zn/NiOOH cells. While Smithrick used relatively high frequencies, Wagner found an optimized low-frequency pulse of 12Hz, with a 9 to 1 rest-to-pulse time ratio. Cells charged with this pulse had better capacity retention and less shape change than those charged at either constant current or constant voltage.

The experimental conditions utilized by Wagner were not as severe as those expected in aplications such as electric vehicles. The cells were cycled at 80% of the rated capacity, which was defined as 68% of the theoretical capacity. Based on the theoretical capacity, the depth-of-discharge was therefore only 54%. In addition, the 10-hour recharge time was relatively long. Therefore, though the reported results show beneficial effects of the pulsed charging, they cannot be readily extended to cells cycled at close to 100% depth of discharge.

Appelt and Juerwicz (23) have investigated the effect of pulsed-current plating conditions on the morphology of electrodeposited zinc. Optimal current ratios were identified.

The aim of this work is to determine the effect of several pulsed charging modes on cell cycle life under conditions similar to those expected for the application of Zn/NiOOH batteries in electric vehicles.

Table 1

Battery Parameters

	Development Goals	Expected Vehicle Performance(1)	Zn/NiOOH Best Results at NBTL Jan. 1981
Specific Energy	56 W-hr/kg (C/3 rate)	100 miles/charge (SAE 227a D Urban Driving)	68 W-hr/kg
Specific Power (at 50% DOD)	104 W/kg (30 sec peak)	Acceleration 0 to 30 mph in 8 sec	131 W/kg
Cycle Life	800 Cycles (80% DOD)	>60,000 miles	179 cycles
Manufacturer Price	70 $/kw-hr	$2,000 battery cost	

(1) Based on DOE/ETV-1 Vehicle with regenerative breaking

REFERENCES

1. S. U. Falk, A. J. Salkind, "Alkaline Storage Batteries", Wiley, New York, 1969.

2. N. P. Yao, C. C. Christianson, F. Hornstra, Proc. of 16th IECEC, Atlanta, 1981, p.641

3. Albert Himy, Proc. of 16th IECEC, Atlanta, 1981, p.645

4. W. G. Sunu, D. N. Bennion, J. Electrochem. Soc., 127, 2007 (1980)

5. T. P. Dirkse, L. A. Vander Lugt, N. A. Hampson, J. Electrochem. Soc., 118, 1606, (1971)

6. J. McBreen, E. J. Cairns, "Adv. Electrochem. & Electrochem. Eng.", 11, Gerisher and Tobias Ed., Wiley, New York, 1978, pp 273-352.

7. T. B. Dirkse, Technical Report No. AFAPL-TR-72-87, Contract No. AF33[615]-3292, Calvin College, Grand Rapids, Mich., Dec 1969.

8. J. McBreen, J. Electrochem. Soc., 119, 1620, (1972)

9. K. W. Choi, D. N. Bennion, J. Newman, J. Electrochem. Soc., 123, 1628, (1976)

10. M. B. Liu, G. M. Cook, N. P. Yao, J. Electrochem. Soc., 128, 1663, (1981)

11. Energy Research Corporation, "Annual Report for 1979 on Research, Development and Demonstration of Nickel/Zinc Batteries for Electric Vehicle Propulsion", Argonne National Laboratory Report No. ANL/OEPM 79-10, (June 1980)

12. R. F. Thornton, E. J. Carlson, J. Electrochem. Soc., 127, 1448, (1980)

13. J. McBreen, Extended Abstracts, The Electrochemical Society, Princeton, NJ, Vol. 77-1, Abstr. No. 356, p. 909.

14. O. C. Wagner, "High Cycle Life, High Energy Density Nickel/Zinc Batteries", Report No. 3, U. S. Army Electronics Research and Development Command, Fort Monmouth, New Jersey, Report No. P50-3C, (February 1980).

15. Gould, Inc., "Annual Report for 1980 on Research, Deveopment, and Demonstration of Nickel/Zinc Batteries for Electric Vehicle Propulsion", Argonne National Laboratory Report No. ANL/OEPM-80-13, (March 1981).

16. EPA Urban Driving Profile, Federal Register, 11/10/70

17. F. Hornstra, E. Berrill, P. Cannon, D. Corp, D. Frederickson, V. Kremesec, W. Lark, C. Swoboda, and C. Webster, "Results of Simulated Driving Profiles in the Testing of Near-Term Electric Vehicle Batteries", Proc. EVC Expo '80 Conf., St. Louis, (May 20-22), Paper EVC No. 8034, Electric Vehicle Council, Washington, D.C. (1980).

18. N. Ibl, F. Leaman Ed., AES 2nd International Symposium on Pulse Plating, American Electroplaters Society (1980).

19. V. V. Romanov, Zhur. Priklad. Khim., 34(12), 2692 (1961).

20. D. N. Bennion, "A Review of Membrane Separators and Zinc/Nickel Oxide Battery Development", Final Report for Argonne National Laboratory, Contract No. 31-109-38-5455 (1980).

21. D. T. Chin, S. Venkatesh, J. Electrochem. Soc., 128, 1439, (1981)

22. J. J. Smithrick, "Effect of Positive Pulse Charge Waveforms on Cycle Life of Nickel-Zinc Cells", DOE-NASA/1044-79/13, (1979).

23. K. Appelt and J. Jurewicz, "Uber den Einfluss des elektrolytischen Abscheidungsprozesses von Zn mittel eines Drei-Komponenten-Impulsstromes auf die Morphologie der Zinkelektrode des Ni/Zn-Akkus," Extended Abstracts of the 29th Meeting of the International Society of Electrochemistry, p. 862 (1978).

CHAPTER 2

DESIGN OF THE MULTIPLE ELECTROCHEMICAL CELL CYCLING SYSTEM

2.1 DESIGN CONSIDERATIONS

The low cost and wide commercial availability of microprocessors make them very powerful tools for use in many applications. When properly configured with memory and input/output peripherals, they provide a convenient means of controlling and monitoring experiments, which can require the accumulation of voluminous data over an extended period of time.

Experiments to evaluate the performance of secondary batteries fall in the above-mentioned category. Indeed, one of the key parameters in the assessment of the practicality of a secondary battery is its durability or cycle life (1). This is determined by repeatedly discharging and recharging the battery under pre-established operating conditions until its capacity falls below a certain level. In order to understand why the battery fails, it is appropriate to monitor cell voltage, the potential of each electrode versus a reference electrode, temperature, and other relevant variables throughout the entire length of the experiment. Data can be gathered with strip-chart recorders or other analog devices, but the accumulated results must then be processed manually, which is at the very least cumbersome if not inaccurate. It is thus

quite natural that, in recent years, much work has been devoted to establish micro or minicomputer automated battery test systems (2,3,4).

One of the objectives of this work has been to design and construct a computerized system that can serve as a tool for electrochemical experiments. It was decided to build the system around an electronic circuit that provides fast and accurate control of the current that flows in an electrochemical cell. A minicomputer is programmed to provide a setpoint for this current and monitor the experiment. Since the analog control function is elementary (current control) it is the computer that has the important task of organizing the experiment: at the cost of effort to program the computer, a maximum system flexibility is derived.

A library of hardware-driving software routines has been written to provide a clean interface between an application program and the external hardware functions (TTL input/output, analog to digital and digital to analog conversion, etc...). These routines can be called as regular subroutines of a separate program, and they make the internal workings of the hardware invisible to the user. These routines permit the programmer to easily write his own application programs in FORTRAN.

The bulk of this work was dedicated to creating a set of application programs aimed at the implementation of a cycle life testing system for electrochemical cells.

The system design is general and can handle testing on various types of electrochemical cells. However, it has so far only been used for the testing of experimental Zn/NiOOH cells, and this discussion is,

therefore, written with that application in mind. The system can, of course, be adapted to control and monitor a wide range of experiments.

2.2 OVERALL SYSTEM OBJECTIVES

A number of basic capabilities of a test system for cycling electrochemical cells are required, and all of them were realized through appropriate hardware and software design, as is described in later parts of this chapter (Sections 2.3 and 2.4) and in the appendices.

2.2.1 SYSTEM SIZE

A common cycling test for Zn/NiOOH cells is to charge at the 5-hour rate and discharge at the 3-hour rate. With short open-circuit periods between successive charges and discharges, one can then accumulate about 3 cycles a day, or 100 cycles a month. This lengthy duration clearly dictates the need to test many cells simultaneously. However, as each cell gradually loses capacity at different rates, it is impossible to keep their cycling sequence synchronized: therefore, a desirable feature of the system is the capability to independently control a maximum number of cells. It was decided to size the system for 8 independent experiments, each capable of driving a maximum of 2 cells connected in series.

2.2.2 TESTING MODES

As mentioned in chapter 1, there are many methods to charge or discharge an electrochemical cell. At times, the cycling mode itself can be the subject of the experiment. The following testing modes were specified:

1. Charging at constant current or at constant voltage.

2. Charging with an externally imposed current profile.

3. Discharging at constant current or at constant power.

4. Discharging with an externally imposed current profile.

5. Discharging with a rapidly fluctuating power profile, such as the EPA urban driving profile (see Appendix G)

It is necessary to select the current, voltage or power levels, the duration of charge and discharge; the open-circuit resting time; and other parameters which characterize the overall cycling regime.

2.2.3 CONTINUOUS OPERATION

The system must operate without interruption. This constraint is necessary for consistency in the comparison of various experiments. For instance, it has been reported that hydroxide concentration gradients (5) and potential gradients (6) in porous Zn electrodes relax upon current interruption (a short-term effect). Longer-term effects, such as self-discharge of electrodes in cells, have also been reported (7). These effects can vary according to the state of charge of the cells at the moment their charge or discharge is interrupted. This phenomenon complicates the interpretation of the cycling results, underscoring the need to avoid unscheduled interruptions. Continuous operation creates several constraints:

1. Since the test regimes of various cells are not synchronized, there must be a convenient means to remove a failed cell from the test loop (or add a new cell to the test loop) without interrupting other experiments. One cannot wait until all the cells are in open circuit.

2. The controlling hardware must operate in a stable manner for long periods of time, and self-heating effects must be minimized since excessive temperature changes can alter the calibration of electronic parts and thereby reduce the accuracy of the measurements.

3. Since many concurrent experiments are monitored by the computer, a huge volume of data must be kept in files on large-capacity storage devices.

2.2.4 CELL PROTECTION

Cells under test must be protected from severe overcharge or over-discharge conditions:

1. One must be able to assign upper and/or lower limits to any of the variables measured (voltages, current, temperature,...), in order to terminate a charge or discharge when appropriate.

2. After a power outage, when the power is restored, the cells must be left in open-circuit mode.

3. In case of a computer program failure or a system crash, the controlling hardware must automatically disconnect the cells.

2.2.5 TIMESHARING SYSTEM

A computerized test system is not practical if the computer is continuously controlling experiments and cannot tolerate interruption. It is important to frequently analyze recorded data, preferably shortly after they have been taken, to enable corrective action to be taken as the experiment progresses. An obvious solution to this problem is to employ a second computer, which has access to the same mass storage device as the computer which controls the experiments. One can then run programs for data retrieval, analysis or graphic display. The second computer could also be used to write new computer programs or improve existing ones. Unfortunately, financial constraints dictated against the purchase of a second machine, and an alternate approach, the installation of a direct link between our computer and the computing facilities on the U. C. Berkeley campus or the Lawrence Berkeley Laboratory could not be easily implemented. It was therefore decided to implement a timesharing system on our computer, which would allow more than one program to run at the same time. This solution is described in detail in the software sections.

2.3 SYSTEM HARDWARE

A battery or electrochemical cell test system can schematically be represented as in Fig. 1.

To be computer-controlled, any such system needs a logic control interface: relays must be switched to connect or disconnect cells and change the current polarity; and a setpoint for the applied current must be provided. It is also necessary to have appropriate circuit elements dissipate the power which the cells deliver on discharge (usually a bank of resistors or diodes), and a monitoring interface for data collection is required.

A cost-sensitive element of these systems is their analog control circuitry. One possible configuration includes a programmable power supply for each independent cell under test. Another configuration could employ a custom-made cycler (3,4), with an integrated power supply and analog and digital control design. In both cases, one cycler per test unit is required.

A cost barrier to testing many cells is the need to purchase many programmable power supplies with acceptable specifications (see below). A cost-effective power delivery configuration was proposed by the Department for Instrument Science and Engineering, at the Lawrence Berkeley Laboratory. Two prototype current controllers were constructed, installed and tested in laboratory operating conditions. These current controllers were designed to operate a multiple number of cells from only two common unregulated power supplies. With a larger system as objective, a cost analysis was made

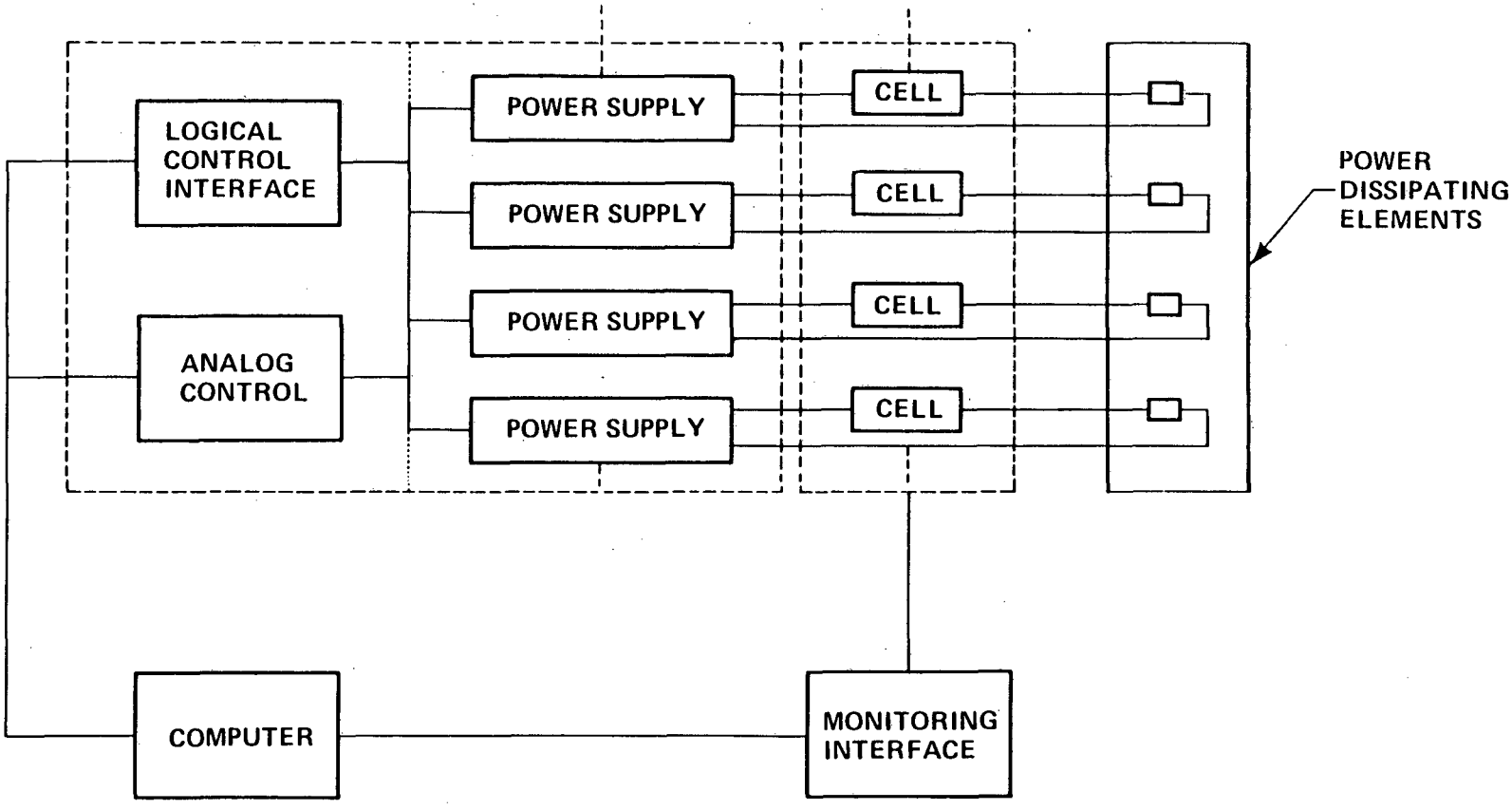


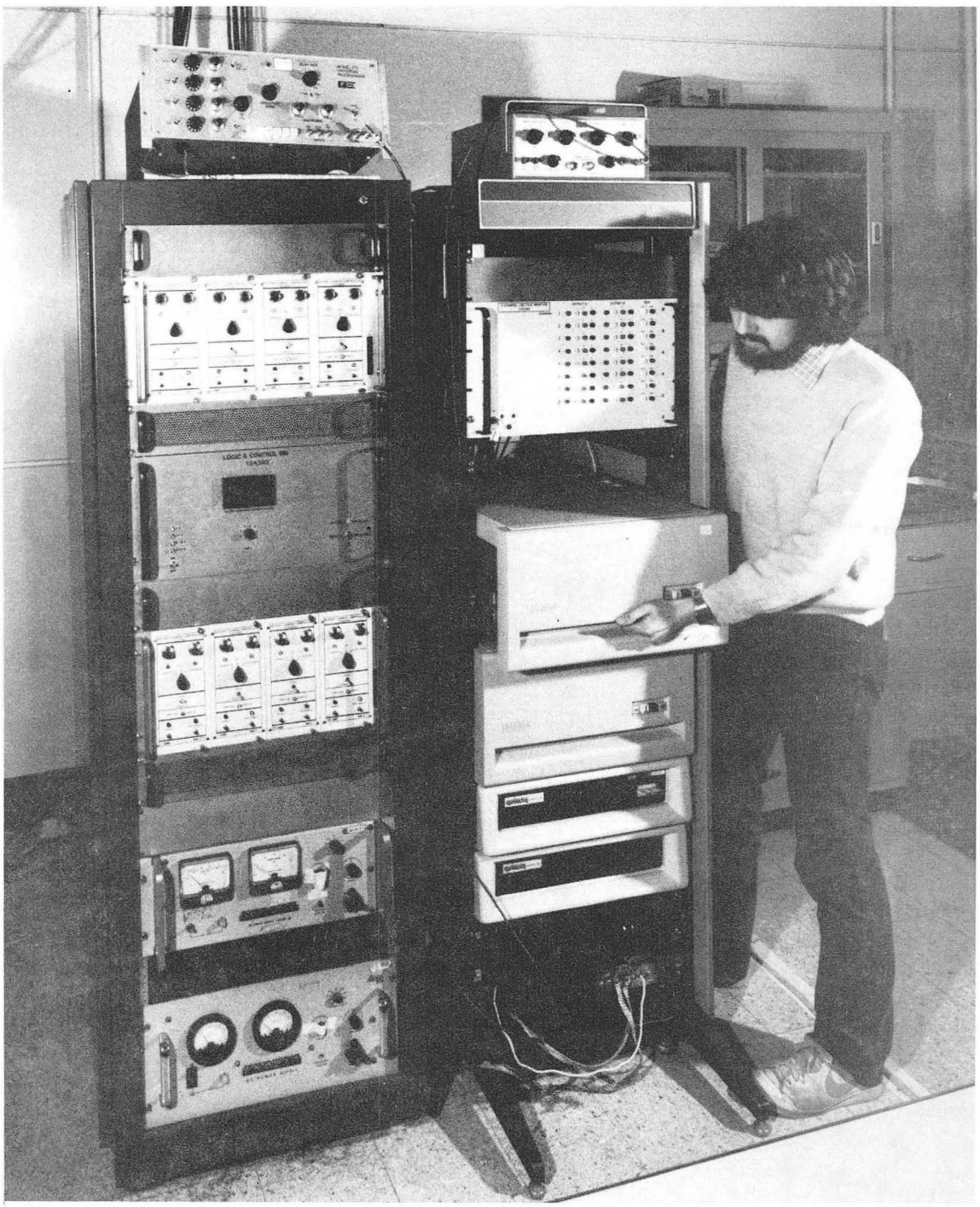
FIGURE 1. Schematic Diagram of an Electrochemical Cell Cycling System.

to compare the prototype design with commercial products. Based on the cost analysis, it was decided to build a system with 8 current controllers, each one dedicated to 2 cells connected in series. The hardware system is described in detail in Appendix A, and a brief summary follows:

> The heart of the hardware system is the current controller, which has an analog circuit combined with a logic circuit. The analog circuit is a feedback control regulator, with a VMOS transistor as linear dissipative control element. The current is regulated at a value proportional to a set-point control voltage which is provided by a local source (for manual control), by the computer, or by a function generator. The computer monitors the cell current, polarity, and voltages. The digital logic circuit switches relays to connect or disconnect a cell, to place it in charge or discharge mode, or to allow the controller to operate in external mode. Common to all controllers is a logic timer circuit which disconnects every cell if the computer program fails.
>
> The system has the necessary flexibility and fast (ms) response time to accommodate a wide variety of cell charge and discharge waveforms. The system control hardware provides 0.025% of full-scale resolution in current settings. Current measurements show less than 0.025% drift of current due to temperature or circuit instabilities in <u>any</u> of the current controllers. Current errors due to

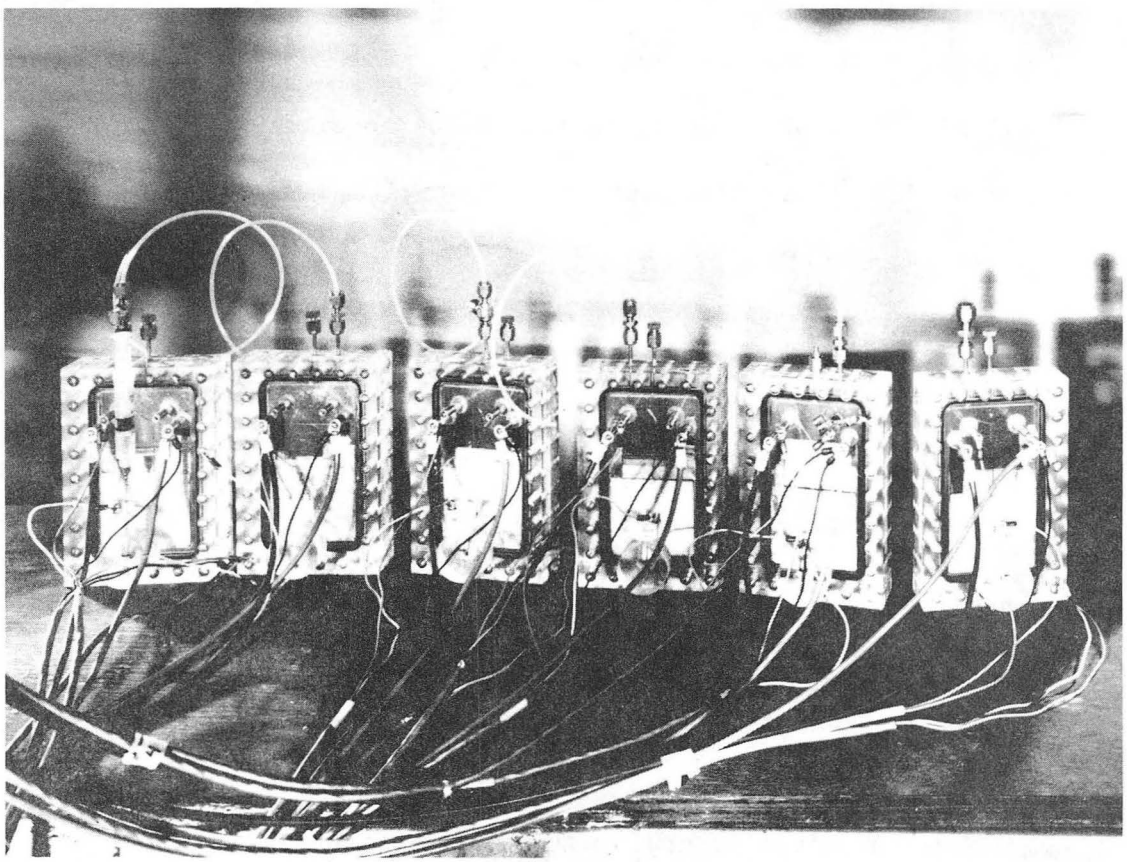
controller cross-talk are less than 0.025% of full-scale within 1 msec after an abrupt current change in any other controller.

Figure 2 is a photograph of the system. Figure 3 is a photograph of Zn/NiOOH cells under test.



CBB 823-2908

FIGURE 2. Eight Current Controller with Logic and Control Bin, Power Supplies, Voltage Monitor Box and Mini-computer System.



CBB 823-2906

Figure 3. Zn/NiOOH Cells Under Test.

2.4 SYSTEM SOFTWARE

The microcomputer used in this system is an LSI-11/23 model from Digital Equipment Corporation (DEC). The operating system is RT-11, also from DEC. An operating system consists of a number of programs that allow the user to perform the following (among other) tasks:

1. Write text files on storage devices (disks, tapes, cassettes,...);
2. Compile programs written in a high-level language (FORTRAN, BASIC,...);
3. Use special subroutine libraries (mathematical, statistical, timer routines,...);
4. Assemble programs written in assembly language; and
5. Run programs, communicate with line-printers and terminals, and open and close data-files.

The three major accomplishments in writing software were:

1. The hardware I/O library and hardware I/O driver.
2. The real-time application program.
3. The data-processing programs.

2.4.1 THE HARDWARE I/O LIBRARY AND HARDWARE I/O DRIVER

2.4.1.1 HARDWARE I/O DEVICES

All hardware input/output devices that are interfaced with the computer are physically connected to memory locations in its core. The operating system recognizes that a section of the memory is thus physically tied to the outside world and calls it the I/O page. Each device typically uses one or more memory locations in the I/O page: one that contains the information to be transferred, and one that contains information about how the transfer is to be carried out (status register). An I/O device is simply an integrated circuit board that is plugged into the computer box. For each device, the addresses of the memory locations used can be set by means of dip-switches or proper wire-wrap configurations, which are located on the printed-circuit boards. Aside from the disks, the terminals, the line-printer and the plotter, the following hardware devices are used (see system hardware, in Appendix A)

1. 2 16-bit parallel I/O cards (input and output).
2. 2 cards with a total of 8 12-bit digital to analog (D/A) channels (output).
3. 4 cards with a total of 64 12-bit analog to digital (A/D) channels (input).
4. A programmable clock.

With the exception of the programmable clock, the output of each of the cards is selectively wired to some part of the analog hardware. Therefore, an action on the hardware is achieved by reading a digital word, writing a digital word, or modifying the bit pattern in a digital word at the correct address.

2.4.1.2 THE HARDWARE I/O LIBRARY

Assembly language routines were written to organize the information transfer to and from the appropriate I/O devices, thereby implementing the following functions:

1. Connect or disconnect a cell (or string of 2 cells in series) to/from a current controller (1st parallel I/O card, output).

2. Switch a controller from/to the charge/discharge mode (1st parallel I/O card, output).

3. Read the status of a controller, that is computer/local (1st parallel I/O card, input).

4. Set a controller in the external or normal mode (2nd parallel I/O card, output).

5. Run a current through a cell (D/A channel, output).

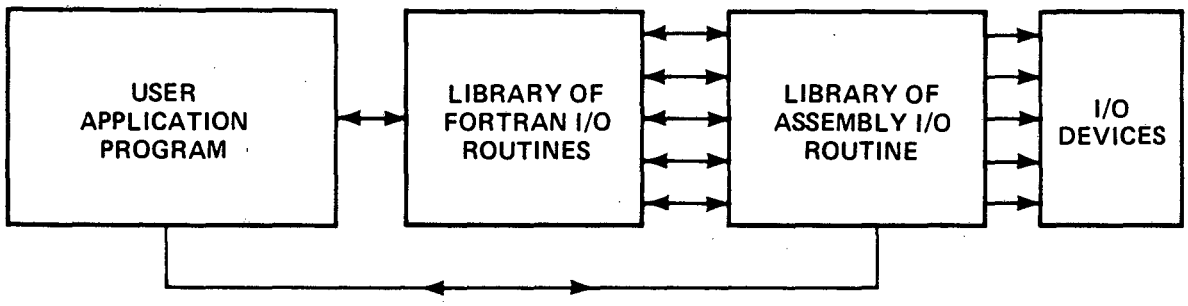
6. Read the current flowing through a cell, the cell voltage, or the potential difference between each of its electrodes and a reference electrode (A/D channels, input).

7. Follow a variable signal and determine its maximum and minimum value over a certain period of time (max. 1 sec) (A/D channel, input, + programmable clock).

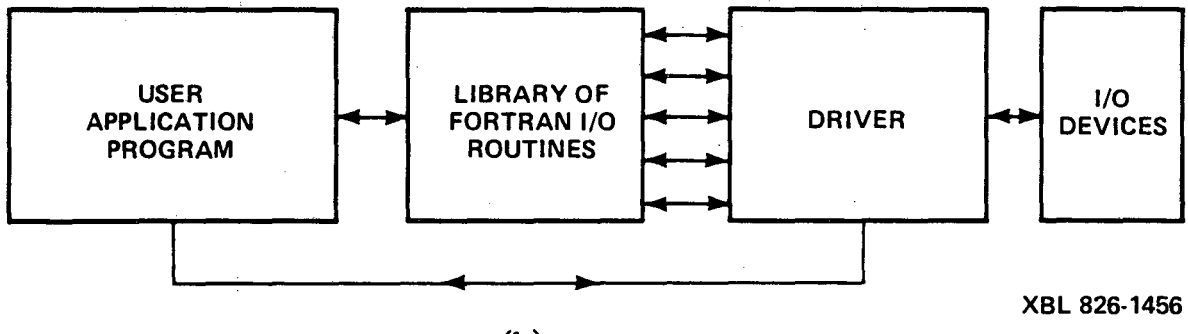
Each of these routines can be called from a FORTRAN user program, with proper specification of input and output parameters. A typical example is a routine that establishes the voltage setpoint of a current controller: the full-scale current rating of the controller (2 A or 10 A) corresponds to the full-scale value of the setpoint voltage (10 V) which in turn corresponds to the highest decimal number that a 12-bit D/A card can output (4095). A smaller current is an integer fraction of 4095. To allow the user to avoid such minor complications, the resolution of which requires some insight into how the computer hardware operates, an additional set of short FORTRAN routines were written; they perform the calculations described above and have options to print messages. For instance, the routine for setting the current is called with convenient input parameters: the current value, the controller number and full-scale current rating. A schematic representation of I/O operations with the I/O library is given in Fig. 4a.

2.4.1.3 THE I/O DRIVER

Appendix B describes the inherent memory restrictions of RT-11, the need to rely on extended memory support, and the procedure to run virtual jobs. Unfortunately, when virtual jobs are run, a major advantage is lost: it is not possible to directly access the I/O page. Therefore, as described in the Appendix B, it was necessary to write an I/O driver. Fig. 5 schematically shows the difference between normal I/O and I/O with a driver.



(a)



(b)

FIGURE 4. (a) Input/Output with FORTRAN and ASSEMBLY language I/O Libraries

(b) Input/Output with FORTRAN I/O Library and I/O Driver.

The original FORTRAN library was modified to allow the user to work with the same FORTRAN calls, thereby insulating the user from the difference between regular and driver-aided I/O. (Fig. 4a is effectively identical to Fig. 4b).

The I/O library and the I/O driver are described in detail in Appendix C.

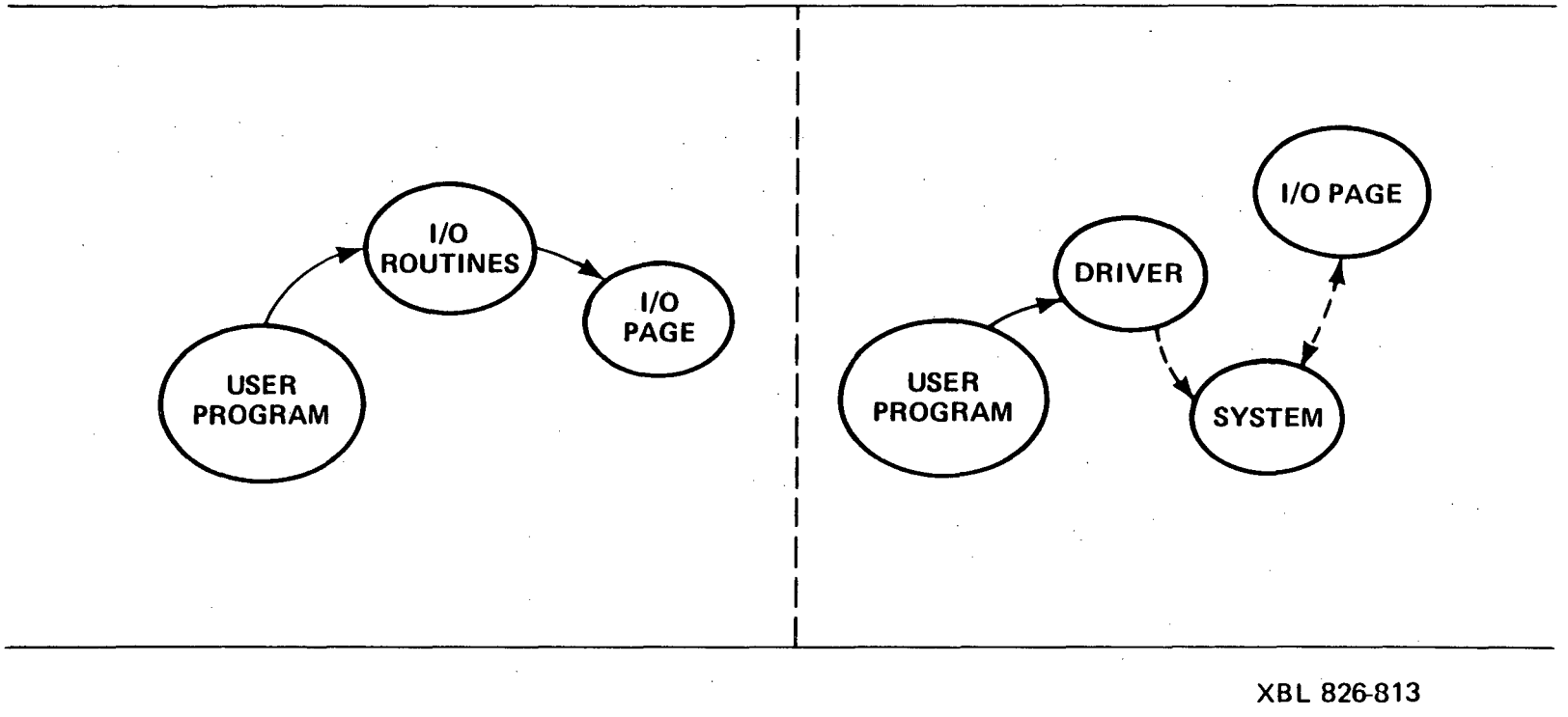


FIGURE 5. Schematic Diagram of Regular and Driver Aided Input/Output

2.4.2 THE REAL-TIME APPLICATION PROGRAM

The REAL-TIME program is the core of the testing system. It is a large program that is resident in the computer memory at all times, and its task is to continuously control and monitor all the cells under test. The program is written in FORTRAN and employs the I/O library described in the previous section. As the tool that controls experiments, the real-time program must be convenient to use, but it must be as general as possible.

1. A wide selection of experimental parameters is needed so that the user is not limited in the variety of experiments he/she wishes to carry out.

2. The inexperienced user must be provided with a straightforward way of initiating or interrupting a test and observing how the test is progressing.

3. For long uninterrupted tests, it is useful to have a way of interactively altering the test regime.

4. In case of a computer-related failure, easy resumption of all tests is of great importance.

2.4.2.1 THE REAL-TIME PROGRAM

Each cell under test requires that certain control tasks be performed at regular time intervals. An action timer is assigned to a particular task (or group of tasks), and it tracks the time of day at which this task must be carried out. The REAL-TIME program 'sits' in a main control loop, continuously updates its internal timer (computer clock), and compares it to the action timers for all cells under test. When it finds one that is less than or equal to its internal timer, the program

updates its internal timer (computer clock), and compares it to the action timers for all cells under test. When it finds one that is less than or equal to its internal timer, the program transfers control to a subroutine which performs the task in question. At the end of the subroutine, a time interval (selectable for some tasks) is added to the internal timer, thereby updating the action timer for that task. For each cell, the following tasks must be performed:

1. Implementation of a current-control algorithm for any of the following charge or discharge modes:
    a. constant current
    b. constant power
    c. constant voltage
    d. pulsating current
2. Taking data, checking for limits exceeded, updating data-taking rate
3. Opening or closing data files, storing data
4. Disconnecting the cell from its controller at the end of a charge or a discharge
5. Connecting the cell with the correct polarity, at the start of a charge or a discharge

6. Checking if the hardware is properly connected (e.g. controller in computer mode)

7. Terminating a test on the cell if appropriate

8. Updating a file with parameters that characterize the cell operation

9. Tracking cycle numbers, date and time

2.4.2.2 FOREGROUND-BACKGROUND; REAL-TIME AND WATCHDOG PROGRAMS

Even with a maximum of 16 cells under test, the REAL-TIME program is not constantly performing control tasks. In fact, most of its time is spent updating its timer and comparing it to the action timers. RT-11 provides means of taking advantage of these inactive periods by allowing the REAL-TIME program to stop its own execution for a particular length of time. When the REAL-TIME program stands idle, another program can execute.

This feature of RT-11 is called Foreground-Background (FB) operation. A foreground (FG) and a background (BG) program share the computer memory: the FG program has priority, but when it suspends its execution, the BG program can run. RT-11 also provides commands that establish a communication path between FG and BG. Messages can be transferred back and forth, much in the same way information is transmitted to and from a terminal or a disk file, by 'read' and 'write' commands.

The REAL-TIME application program consists of 2 programs: the REAL-TIME program (in FG) and a WATCHDOG program (in BG). The reason for this is related to the way input is directed to a program. A 'READ from terminal' statement in a program makes the execution stop until an input is typed on the terminal keyboard. This is not feasible in the context of a REAL-TIME program, because it would mean that all control tasks must wait until all the input has been entered. A convenient way of handling this problem is to have the WATCHDOG program send codified messages to the REAL-TIME program, upon which the latter takes the appropriate action.

There are two CRT terminals attached to the system, as illustrated in Fig. 6. The main terminal is dedicated to the BG, and it allows input of commands and bidirectional information flow with the WATCHDOG program. The second terminal is dedicated to the FG; it does not allow input of commands, and information flow is in one direction only. In case of an emergency, the REAL-TIME program can be interrupted by typing ^C^C at the FG terminal.

As illustrated in Fig. 6, a command to run the REAL-TIME program must be typed on the BG terminal. The program starts, and it just sits in the control loop, typing the time on the FG terminal, because it has no other task to perform. When the WATCHDOG program runs, a set of questions appear on the BG terminal, pertaining to the various actions that can be initiated. These are:

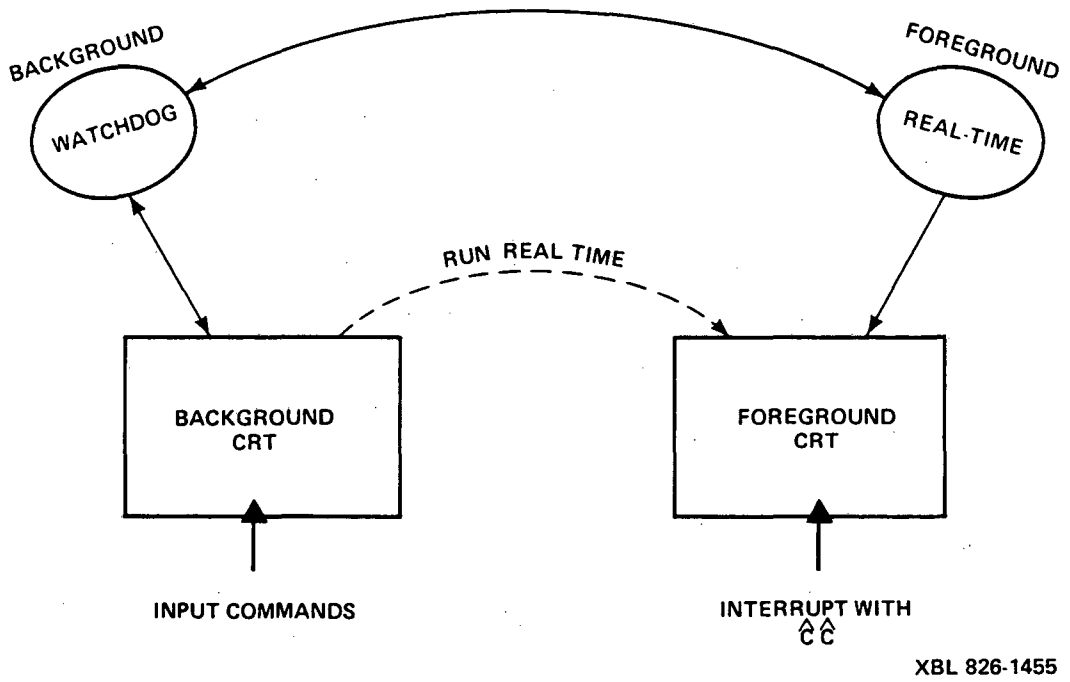


FIGURE 6. Foreground-Background Terminal Configuration.

1. Initialization of a set of parameters that completely define the manner in which a cell will run, and getting a display on the CRT or the line-printer.

2. Modification of some cell operating parameters, such as data-taking rate, current, maximum A-hrs to pass on charge or discharge (but no major changes such as the control mode).

3. Getting a display of the current or newly updated parameters on the CRT or the line-printer.

4. Starting a test on a cell or a series string of cells. There are 2 options:
    a. start it with initialized parameters
    b. start it with present parameters (only if cell was already running)

5. Stopping a test on a cell or a series string of cells. There are 2 options:
    a. stop after a time interval Δt
    b. stop after a certain cycle number, at the end of a charge or discharge

All actions are implemented through a handshake procedure between FG and BG, described in detail in Appendix D. The message-handling components of the REAL-TIME and WATCHDOG programs automatically abort procedures when gross mistakes are detected, such as sending messages to start a cell which is already running, attempting to start a cell

by itself when it is physically connected to another, etc. The user can also abort any procedure previously initiated. When a cell is started, the REAL-TIME program directs some output to the FG CRT at regular time intervals: A-hr count, measurements, and flags that indicate how smoothly the current-control procedure is implemented. When no further action is required, the WATCHDOG program can be terminated without harm to the REAL-TIME program. One can then run another program in the BG, as described in section 2.4.3. At any time, the WATCHDOG program can be re-started and the communication link is automatically established.

A software module interconnection block diagram of the test system is shown in Fig. 7.

The global structure of the REAL-TIME and WATCHDOG programs is quite complex. Detailed descriptions are given in Appendices D and E. However, in order to get a quick overview of how to use the programs the reader is referred to the following sections of this text:

D.1 : describes the cell parameters

D.2 : describes the parameter files

D.3 : describes the data-file structure

E.1 : describes the cycling regimes

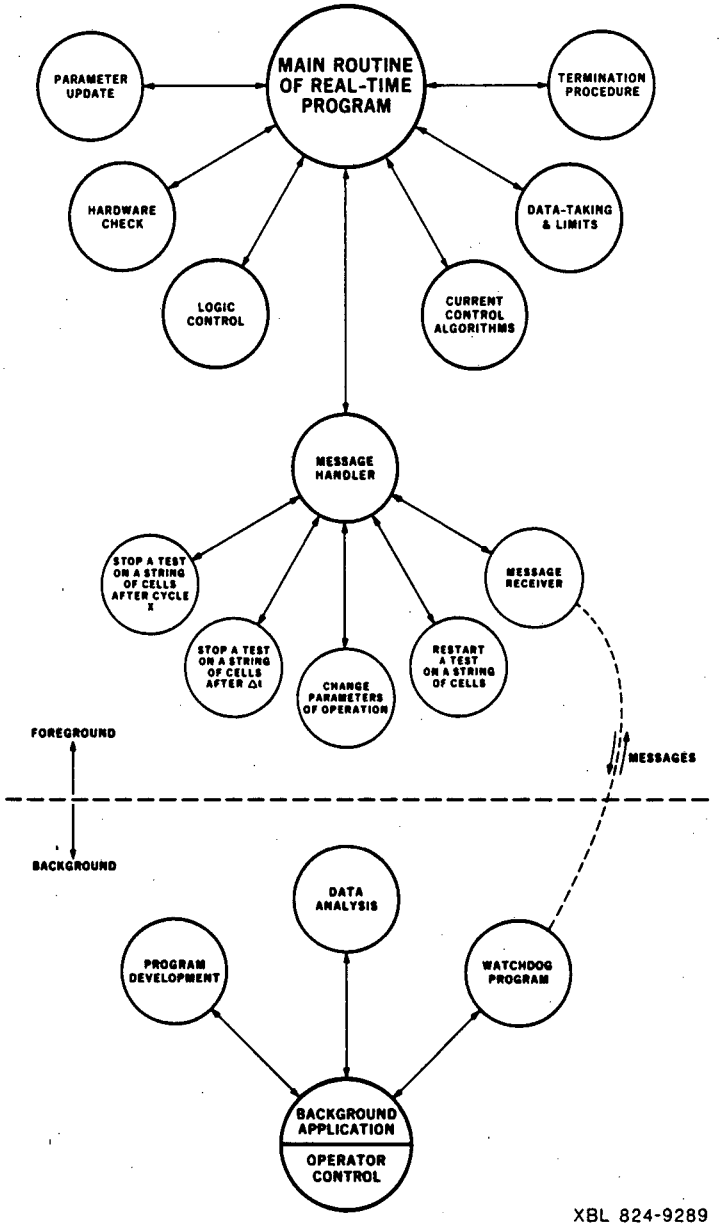


FIGURE 7. Multiple Electrochemical Cell Test System: Software Modules Interconnection Block Diagram.

E.2 : describes the functions of the WATCHDOG program

E.3 : gives useful hints on the use of the programs

F.1 : describes the use of scratch-data-files and archive files

2.4.2.3 SAFETY, COMPUTER AND PROGRAM CRASHES, RE-START

All cells are controlled by one program, so special precautions must be taken to prevent failures.

1. Instructions that can cause fatal FORTRAN errors must not be allowed to execute if an abnormal condition occurs, such as an integer division with a very small denominator. If no obvious correction is possible, the affected cell must be disconnected.

2. One has to anticipate hardware problems, such as an occasional I/O error on a data transfer to a disk.

3. There is no standby power provided for the computer.

The structure of the REAL-TIME program allows for easy recovery from a crash: indeed, the operational parameters for each cell are saved on a file and continuously updated, so that in case of an interruption, a cell can be re-started from the point at which it stopped.

2.4.3 DATA PROCESSING PROGRAMS

As mentioned in section 2.4.2.2., when the WATCHDOG program is not running, another program can run in the background. Such a program can be an RT-11 utility program; i.e., the Editor or the FORTRAN compiler. It can also be another real-time program (provided it fits in memory with the main program) that uses the I/O library, or any other application program (such as a data-processing program).

A schematic overall view of the data-storage and retrieval organization is shown in Fig. 8.

Raw data are stored on separate scratch files for each cell. A program decodes and displays the contents of a scratch file on the CRT or line printer. It also can search for a cycle in the file. Another program decodes the raw data and stores it in well-organized format in an archive file. This file contains the name of the cell and a table of all the cycles stored and their location in the file. Each cycle has a header which contains A-hr, W-hr and average voltage information, as well as the number of variables for each data-point and their identification (voltage, current,..), the number of charge and discharge data-points, etc. An archive file display program and various plotting programs are described elsewhere (8). Data-processing programs are described in detail in Appendix F.

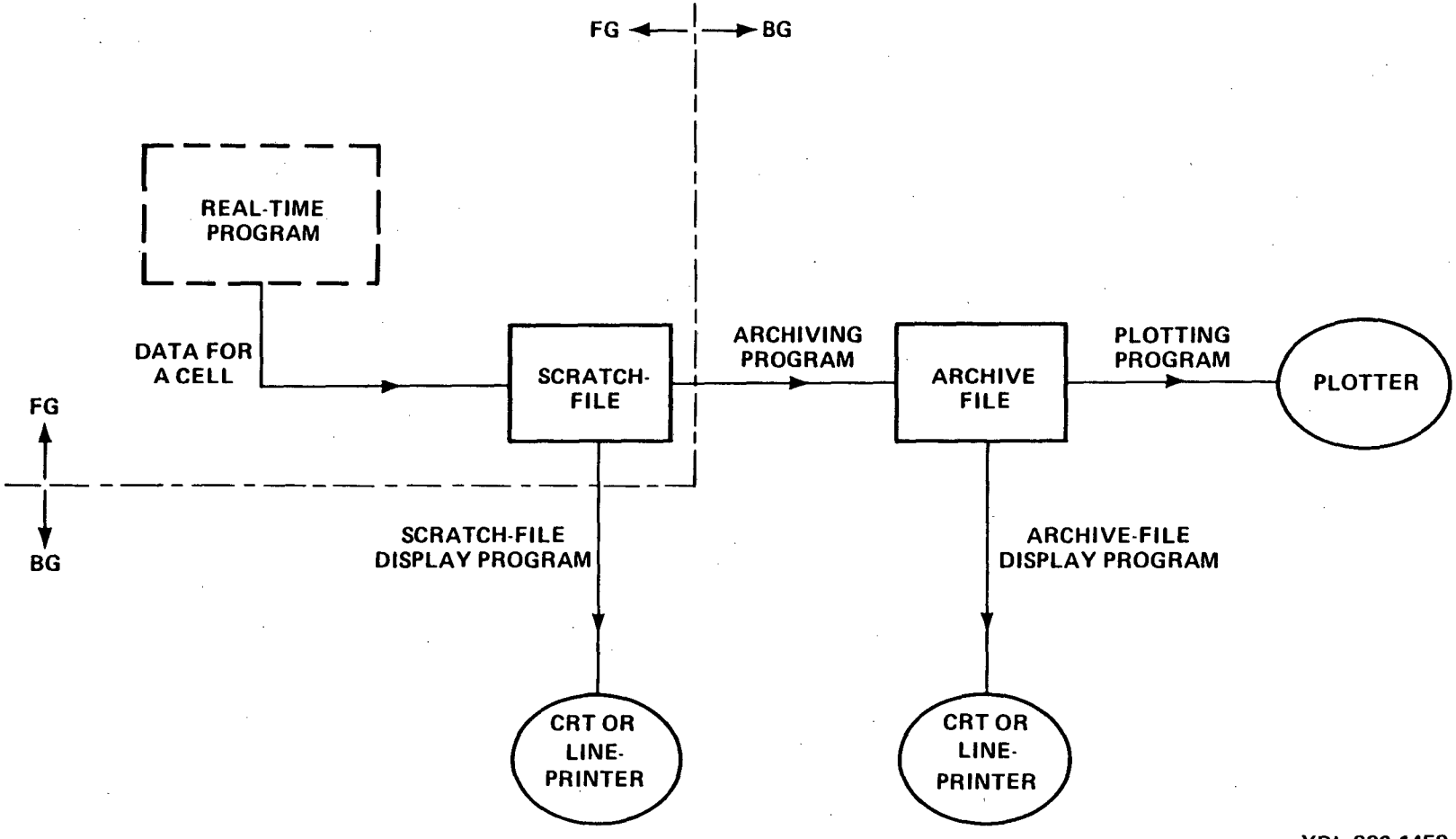


FIGURE 8. Data Reduction Software Diagram.

REFERENCES

1. J. L. Hartman, E. J. Cairns and E. H. Hietbrink, "Electric Vehicles Challenge Battery Technology", proceedings of the 5th Energy Technology Conference, Washington D.C, February 1978.

2. R. P. Hollandsworth and G. B. Adams, "The Role of Microcomputers in Secondary Battery Development and Testing", U.S. DOE report No. PO-6527609, Lockheed Missiles and Space Corporation, December 1979.

3. V. C. Jaswa, "Programmable Battery Test Cyclers", GM report, GM-3065, August 1979.

4. ANL, "The National Battery Test Laboratory", Design Report, ANL/OEPM-77-4, (1977).

5. T. Katan and P. J. Carlen, "Perturbation and Relaxation of Concentration Gradients in Porous Zinc Electrodes", Electrochem. Soc. Symp., Montreal, 1982.

6. J. McBreen, "Zinc Electrode Shape Change in Secondary Cells", J. Electrochem. Soc., 119, 1620 (1972).

7. S. U. Falk, A. J. Salkind, "Alkaline Storage Batteries", p.553, Wiley, New York, 1969.

8. J. Nichols, M.S. Thesis, UCB (1982).

CHAPTER 3

EXPERIMENTAL PROCEDURES

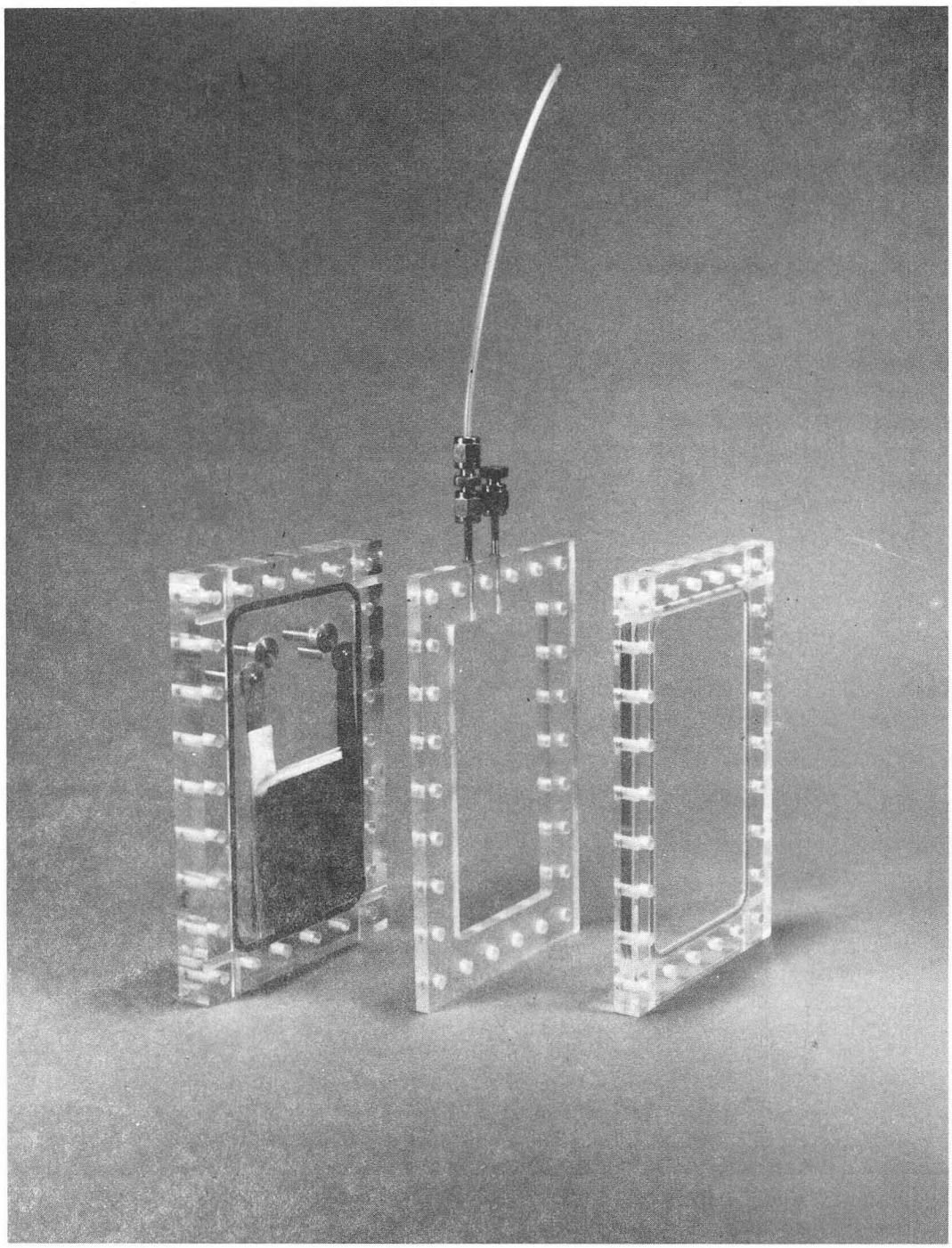
3.1 CELL CONSTRUCTION

3.1.1 CELL CASES AND ELECTRODE PACK

Figures 1 and 2 show one of the cell cases used for experiments. The cases were constructed from lucite on the basis of a design described elsewhere (1).

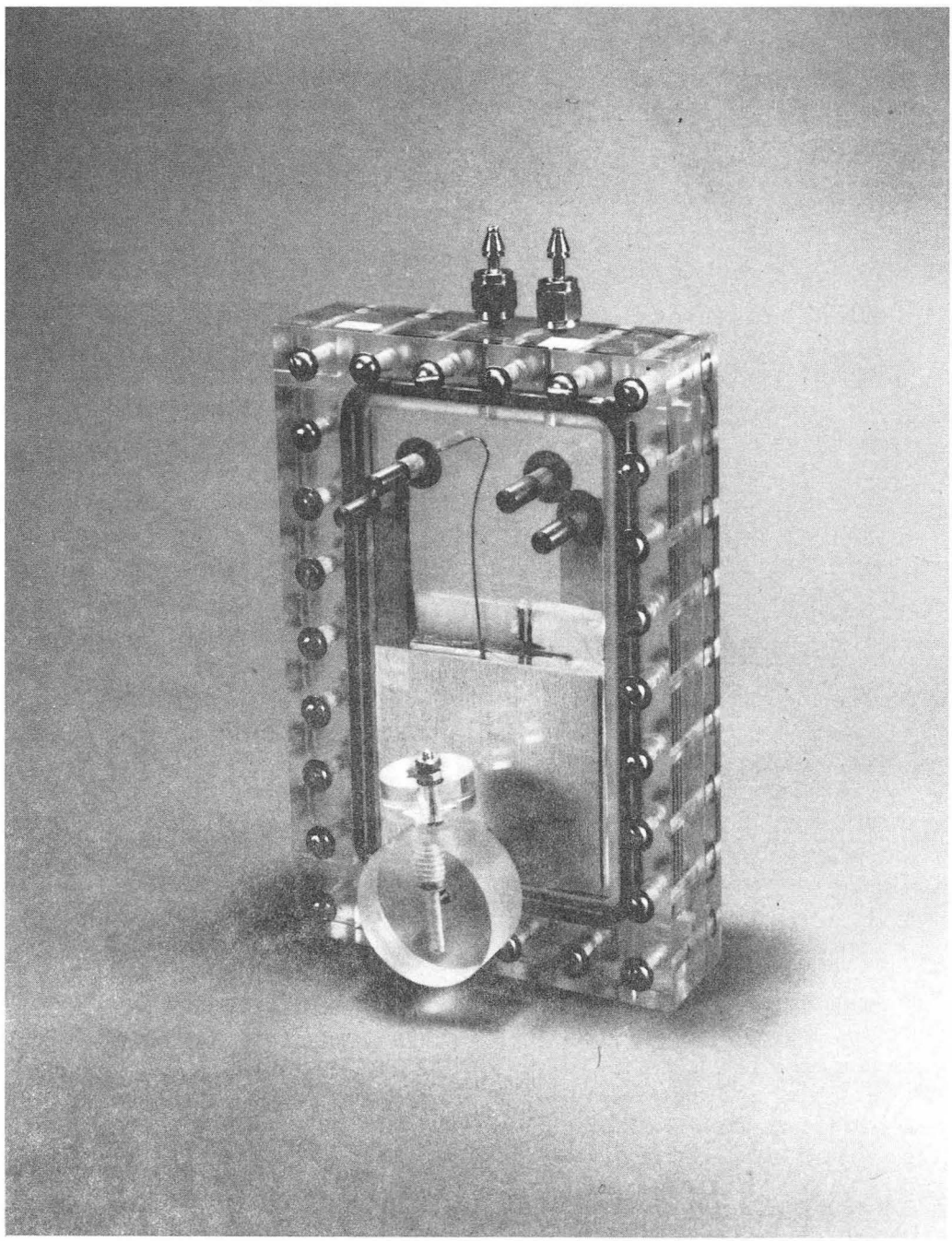
The central portion of the cell case determines the space in which the electrode pack must fit. After the electrode pack is assembled, it is placed between two lead plates with plane parallel faces. A brick is then placed on top of this assembly, to apply an appropriate load of 4 $lb/inch^2$, and the thickness of the cell pack is measured under that pressure. The difference between this thickness and the central portion of the case defines the thickness of a spacer that is subsequently added to the cell pack. The spacer can be machined from lucite or constructed from a stack of polyethylene sheets of various thicknesses.

The complete electrode pack consists of a Zn electrode surrounded by a wick, enclosed with a separator wrap, placed between two NiOOH electrodes, each of which is surrounded by a wick.



CBB 825-4772

FIGURE 1. Zn/NiOOH Cell



CBB 825-4774

FIGURE 2. Zn/NiOOH Cell

3.1.2 Zn ELECTRODES

The Zn electrodes were generously supplied by Energy Research Corporation (2). Each electrode was X-rayed, and only the most uniform were selected for experiments. The specifications follow:

Zn mix	: 95% ZnO, 2% PbO, 3% Teflon
Current collector	: 0.063mm thick Cu foil, 0.046mm thick lead plate
Silver tabs	: 0.127mm thick, 4.3cm x 0.9cm
Dimensions	: 6.2cm x 7.0cm, 0.61mm thick
Capacity	: on each face, 16.7 ± 1.3 mA-hr/cm^2 (this capacity is calculated as [mass ZnO (gr) x 0.659 A-hr/grZnO x 1/3 design utilization] for the two faces of the electrode)
Porosity	: 60%, based on Zn

A possible problem with the use of a foil current-collector (rather than a mesh type) is that the Zn material on one side of the electrode is electrochemically inaccessible from the other side. The electrode can then have two faces acting independently from each other.

3.1.3 NiOOH ELECTRODES

The NiOOH electrodes were fabricated by Eagle-Picher Industries (3) according to the following specifications:

Sintered Ni plaque,	chemically impregnated with $Ni(OH)_2$
Current collector	: Ni mesh
Nickel tabs	: 0.127mm thick, 4.3cm x 0.9cm
Dimensions	: 6.2cm x 7.0cm, 0.48mm thickness

Capacity : 16.7 ± 0.1 mA-hr/cm^2 (specified)

Two NiOOH electrodes are used in each cell pack. The Ni tabs which conduct the current flowing in the cell have a non-negligible resistance; the potential of the Ni electrodes, measured somewhere along the tabs, would therefore include a voltage drop in the tab. For that reason, a Ni wire was spotwelded to the face of one of the electrodes to provide a separate voltage measurement, as shown in Fig. 2.

3.1.4 ELECTROLYTE

The electrolyte composition was 31wt% KOH (Baker Chemical Co., major impurity 0.04% K_2CO_3) and 1wt% LiOH (MCB Co.). An excess amount of ZnO (in excess of saturation (8,9)), was added, and the solution was then stirred for one week.

3.1.5 SEPARATOR AND WICKS

The wick surrounding the Zn electrode consisted of a layer of Aldex paper. Its purpose is to keep the electrode together and to supply it with a reserve of electrolyte.

The separator wrap consisted of the following layers (listed beginning with the layer closest to the Zn electrode):

> 1 layer of microporous polyproplyene, Celgard 3500 (4), used as a protective layer with low resistivity and good mass-transport properties.

> 1 layer of cellophane PUDO-193 (5) : it has a high absorbency for electrolyte and a good retention of dissolved Zn.

> 1 layer nickelized microporous polypropylene, Celgard K317 (4), with the nickelized face adjacent to the Zn electrode. The function of this separator layer is to act as a barrier for Zn dendrite growth by creating a local electrochemical couple that discharges a dendrite while evolving hydrogen on a Ni grain.

> 1 layer of microporous polypropylene, Celgard 3500.

The benefits of using this separator system are described by O. Wagner (6).

The wicks used adjacent to the Ni electrodes were fabricated from non-woven polyamide Pellon #2502K4 (10). Their purpose is to keep the Ni electrodes supplied with electrolyte through capillary action.

3.1.6 REFERENCE ELECTRODE ; FILLING THE CELL ; VENTING

The reference electrode consists of a Pt wire dipped in Hg/HgO in equilibrium with 31% KOH. It is mounted on a small lucite cell that screws into the special compartment on one side of the cell case, as shown in Figure 2. In the original design, a small-diameter hole extended from that compartment, protruded through the cell wall, and emerged at the outer side of the electrode pack. That arrangement provides a potential indicative of the solution at the side of the Ni electrode facing away from the Zn electrode. Therefore, it does not truly reflect conditions near the Zn electrode. To circumvent this limitation, a length of 0.5mm diameter polyethlyene tubing was extended

through the small-diameter hole and through a small hole drilled in one of the Ni electrodes and terminated between the Ni and Zn electrodes. The tubing opening was physically separated from the Zn electrode by the separator wrap; one must then expect to see an increased resistance as a discharge progresses and the separator becomes depleted in OH^-, and impregnated with dissolved Zn (as $Zn(OH)_4^=$).

The small-diameter tubes with fittings at the top of the cell case are employed to fill the cell and to vent gases produced during cycling. During cell operation, one tube is closed and the other is connected to a manifold through which CO_2-free air is supplied. This prevents CO_2 absorption from the air by the electrolyte.

3.2 CELL SPECIFICATIONS

A series of cells were assembled and identified as PWR1 to PWR8. Particular specifications are given in Table 1. The design capacity is based on one-third of the total ZnO capacity initially present in the negative electrodes. The theoretical capacity of the NiOOH electrode is calculated on the basis of 1 electron per Ni atom in the active material, specified as 1.450 ± 0.01 A-hr (counting both Ni electrodes). The original intention was to attain equal design and theoretical capacities, or a total Zn theoretical capacity of 3.0 times the theoretical NiOOH capacity. This was not precisely realized, due to disparities in the Zn electrode capacities, and the actual capacity ratios were 2.99±0.22.

3.3 OPERATING CONDITIONS

3.3.1 FORMATION

Each cell underwent three formation cycles; each was charged for 20 hours at the 10-hour rate (amperes = design A-hr/10 hr) and then discharged at the 3-hour rate (amperes = design A-hr/3 hr) until the cell voltage reached zero. During the second half of the charge, the Ni electrodes evolved oxygen, and during the second half of the discharge, they evolved hydrogen. Water was subsequently added to compensate for losses. During the third discharge, the current was gradually reduced to a small value (~50mA), in order to remove the remaining capacity of the negative electrode before reaching the 0 V limit. Finally, the cell was charged for 20 hours at the 10-hour rate, to establish the metallic Zn reserve, and then was placed on a regular cycling test.

T A B L E 1

CELL CONSTRUCTION

CELL	DESIGN CAPACITY A-hr	THEORETICAL Zn CAPACITY/ THEORETICAL NiOOH CAPACITY	gr ELECTROLYTE/ A-hr
PWR1	1.523	3.15	8.1
PWR2	1.511	3.12	7.7
PWR3	1.488	3.00	8.5
PWR4	1.519	3.14	8.8
PWR5	1.551	3.21	7.1
PWR6	1.339	2.77	8.6
PWR7	1.425	2.95	7.1
PWR8	1.455	3.01	7.3

During the formation process, the capacity of the Ni electrodes generally increases since they are overcharged by about 100%. When regular cycling starts, the cell is effectively cycled at the NiOOH electrode capacity, since there is an excess of Zn metal to accommodate discharge to a depth larger than the design capacity. At a later stage of cycling, the Zn electrode may limit the cell capacity.

3.3.2 CYCLING CONDITIONS

The cells were tested for the effects of various charge profiles on cycle life. All were cycled with the same conditions on discharge, i.e. at constant power, with a cutoff voltage of 1.0 V. The power level for each cell was based on a discharge time of 2.5 hr, and on the Ni capacity and the average cell voltage during the last formation cycle. Thus, the cells were effectively cycled at 100% depth of discharge.

The charge modes compared were constant current versus various pulsed-current profiles. The average current on charge was based on the charge time and the NiOOH electrode capacity at the last formation cycle. Open-circuit periods of 30 minutes were provided at the end of each half-cycle.

A summary of the cycling conditions is provided in Table 2.

TABLE 2

CYCLING CONDITIONS

Cell	PWR1	PWR5	PWR3	PWR8	PWR6	PWR7	PWR2	PWR4
FREQUENCY (Hz)	CONSTANT CURRENT	60	8.3	8.3	8.3	8.3	10	10
ON-TIME (msec)	CONSTANT CURRENT	8.3	30	30	30	30	10	10
OFF-TIME (msec)	CONSTANT CURRENT	8.3	90	90	90	90	90	90
OFF-TIME/ ON-TIME	CONSTANT CURRENT	1	3	3	3	3	9	9
CHARGE TIME (hours)	4.5	4.5	4.5	4.5	2.25	2.25	4.5	4.5
AVERAGE CURRENT (A)	0.358	0.366	0.336	0.336	0.653	0.653	0.352	0.352
PEAK CURRENT DENSITY (mA/cm^2)*	4.1	8.4	15.5	15.5	30.1	30.1	40.6	40.6
POWER ON DISCHAGE (W)	1.062	1.070	0.988	0.988	0.965	0.965	1.044	1.044

* Peak current densitites are calculated with respect to the 2 faces of the Zn-electrode (86.8 cm^2 total area)

The NiOOH electrodes are known to operate at less than 100% efficiency: on charge they evolve oxygen. Examination of the equilibrium potentials indicates that there is approximately a 100 mV driving force for this parasitic reaction.

It is thus necessary to overcharge the NiOOH electrodes in order to enable them to retain their capacity. This is implemented in the computer program by means of an adjustable factor multiplied by the number of coulombs passed on the previous discharge to define the coulombs to be passed on charge. This factor is usually adjusted as cycling proceeds, to maintain the coulombic efficiency (A-hrs OUT/A-hrs IN) approximately equal to the inverse of 1.0 plus the fractional overcharge. In that way, the NiOOH electrode capacity remains constant.

3.3.3 STATE OF CHARGE — BALANCING OF A CELL

Tracking of the state of charge of the Zn electrode is necessary to insure that it is cycled with a sufficient reserve in Zn and ZnO material. This condition, which is achieved at the onset of regular cycling, is illustrated in Fig. 3a. The Zn electrode is exercised over one third of its total capacity. On discharge, Zn metal sufficient to avoid polarization is required for capacity maintenance. It is also needed to provide a current conduction path in the electrode, since ZnO is a poor conductor, and to guarantee the existence of enough nucleation sites at the initiation of charge. At the end of the charge, enough reducible ZnO material is needed to avoid severe mass-transfer limiting conditions and dendrite growth (7).

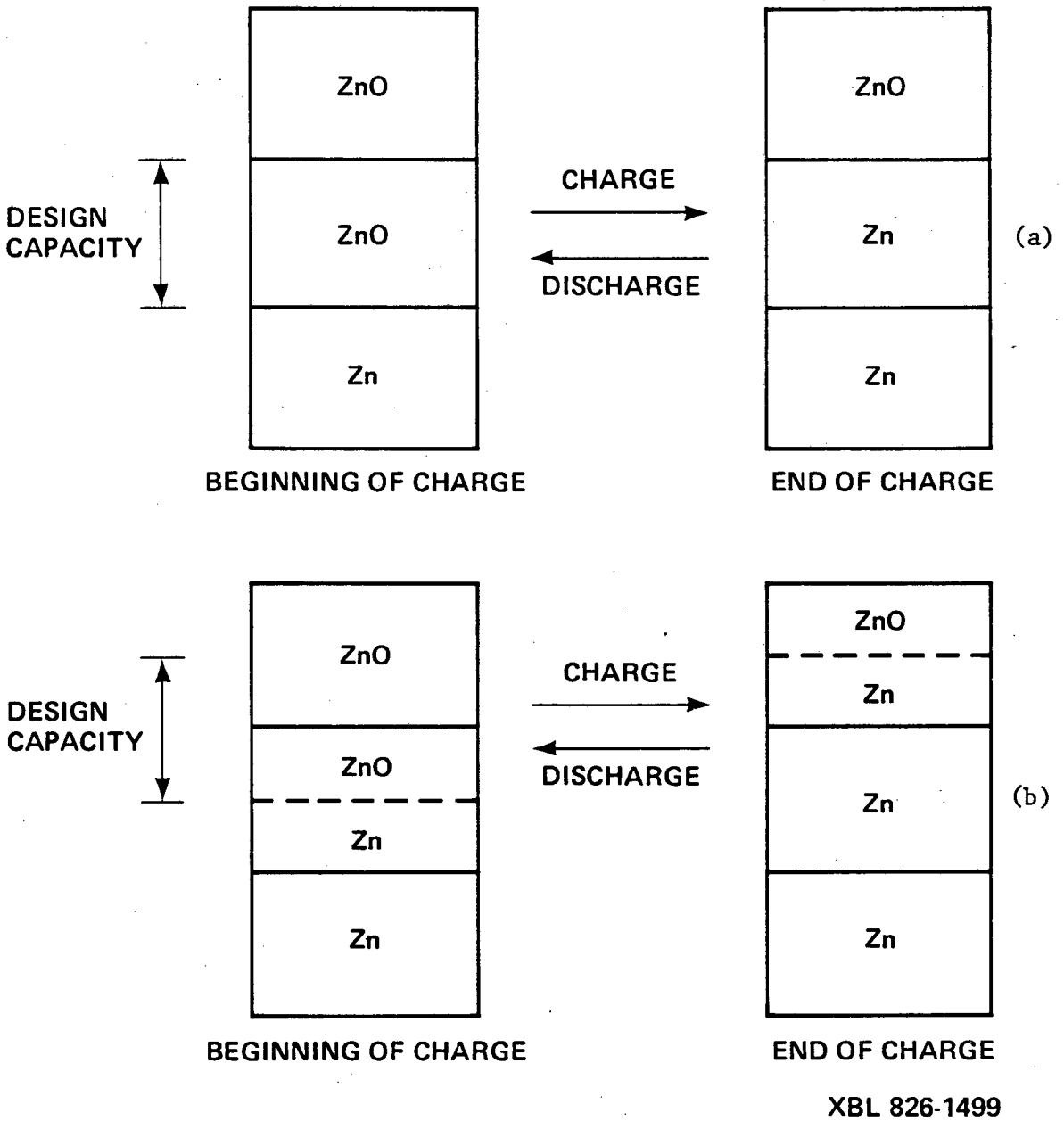


FIGURE 3. Zn and ZnO Reserves in a Cycled Zn electrode.

(a) Beginning of cell life

(b) Reduction of ZnO reserve capacity after extended cycling

With lead added as a hydrogen-suppressing element in the Zn electrode mix, the Zn electrode usually operates more efficiently than the NiOOH electrode. As a cell is overcharged (see Section 3.3.2), the ZnO reserve gradually decreases, as illustrated in Fig. 3b.

After a number of cycles, there is inadequate ZnO reserve at the end of a charge, and one reaches a condition where dendrite growth is favored (7). To avoid this problem, the cell is completely discharged, i.e. until all the Zn metal on the electrode is converted to ZnO. Then, the Zn reserve is formed again, and regular cycling is resumed. Note that this problem does not arise in the operation of sealed cells, where as the oxygen pressure builds up, it recombines with the Zn in the negative electrode and thereby maintains a sufficiently large ZnO reserve.

Several factors made it difficult to keep track of the state of charge of the Zn electrodes.

- The amount of hydrogen evolved on charge was unaccounted for. Some cells were charged with high current density pulses, so the quantity of vented H_2 might be appreciable.

- The amount of Zn that may lose electrical contact with the current collector is not measured. A Zn dendrite can grow through the first layer of separator, discharge at the nickelized separator (or stop growing, due to blockage of a nucleation site when the current is turned off), and at a later time become disconnected at its base. For some cells analyzed after the test, the separator wrap was found to contain some Zn material.

> The amount of Zn that gets electrically isolated within an electrode. Regions of Zn may become surrounded by poorly conductive ZnO and thus become unable to take part in the reaction.

In most cases, when a cell was completely discharged, it was not possible to recover all of the original capacity. It was not possible to determine how the above effects had combined to create that situation, so there was some uncertainty about the amounts of Zn and ZnO reserves at the end of an experiment.

3.3.4 CYCLING CELLS IN SERIES

Cycling cells in series is not a major problem, unless the capacity of one cell differs substantially from that of its companion.

3.3.5 POTENTIAL MEASUREMENTS

The electrical connections extending from the electrode terminals are not impedance-matched with lines connected to the reference electrodes. When a cell is charged with pulsed current, the reference electrode responds sightly more slowly to the pulse, so the differential electrode versus reference signal is actually a deformed pulse. Unfortunately, the software-based measurement technique used for pulsed charges was not sufficiently sophisticated to accommodate the problem, so no valid measurements (versus the reference electrode) could be taken during pulsing. Voltage measurements that were taken included:

1) Cell voltages at current-ON and current-OFF on a pulsed charge.

2) Cell voltages and electrode potentials versus reference on a constant-current charge.

3) idem, on a constant-power discharge.

4) idem, on open circuit.

The measurement of the Zn electrode potential versus the reference, made immediately after interruption of the current at the end of charge, provides useful information. It contains most of the concentration overpotential and gives some indication of the mass-transfer conditions at the end of charge. When the potential reached 1.4V (negative) versus Hg/HgO, i.e. about 75mV overpotential, it was probable that the Zn electrode was operating with a low ZnO reserve.

REFERENCES

1. J. Nichols, M. S. Thesis, UCB (1982)

2. Energy Research Corporation, Danbury, Connecticut

3. Eagle-Pitcher Industries, Colorado Springs, Colorado

4. Celanese, Summit, New Jersey

5. E. I. DuPont de Nemours, Inc., Wilmington, Delaware

6. O. C. Wagner, "High Cycle Life, High Energy Density Nickel/Zinc Batteries", Report No. 3, U. S. Army Electronics Research and Development Command, Fort Monmouth, New Jersey, Report No. P50-3C, February 1980.

7. W. G. Sunu and D. N. Bennion, J. Electrochem. Soc., 127, 2007, 1980.

8. T. P. Dirkse, J. Electrochem. Soc., 106, 154, (1959).

9. R. F. Thornton and E. T. Carlson, J. Electrochem. Soc., 127, 1448 (1980).

10. Pellon Corporation, Chelmsford, Massachusetts.

CHAPTER 4

ANALYSIS OF RESULTS

4.1 INTRODUCTION

The experiments described in this chapter were carried out with the purpose of analyzing the effects of three parameters of a pulse-charging profile:

- the off-time
- the off-time/on-time ratio
- the peak current density

As explained in Chapter 1, a high current density can increase the number of nucleation sites for the charge reaction on the Zn electrode, thereby reducing dendrite initiation. However, an unwanted side effect expected is inefficient operation of the electrode. A sufficiently long off-time improves the mass-transfer conditions for the reaction by allowing zinc ions to diffuse into the depleted diffusion layer, thereby reducing dendrite growth.

In most of the experiments the charge time was left at 4.5 hours: thus, it was not possible to vary only one of the parameters named above and keep the other two constant.

Section 4.2 of this chapter compares a cell charged at constant current and two cells charged with a pulsed current. The parameters varied were the off/on ratio and the current density; the parameter kept constant was the off-time. Section 4.3 compares the results of two

duplicate cells. Section 4.4 compares a cell charged at constant current with one charged with a 60 Hz pulsed current: the purpose of this comparison was to demonstrate that a short off-time does not improve cell cycle life. Finally, Section 4.5 gives the results for two duplicate cells, charged at a regime intermediate between the two regimes described in Section 4.2. Taken together, Sections 4.2 and 4.5 point to the effect of an increasing peak current density, for a constant off-time.

A summary of pertinent results is given in Table 1. The listed capacity loss rates are seen to vary by an order of magnitude, ranging from 0.21 to 2.72 %/cycle. Notation Z or N, seen in the 6th column, indicates which of the electrodes (Zn or Ni) was limiting the capacity of the cell at the end of the test. In the discussion that follows, the notations PWR1, PWR2, etc. will include designations of on/off times and peak current densities: ms/ms, $mA\text{-}cm^{-2}$.

Cell PWR3 (30/90,15) displayed the most stable performance. As seen from Table 1, it was the only cell which retained a capacity higher than 75% over its entire test period. It was also the only cell in which the capacity decline matched the reduction of active area. The cells charged in the other modes exhibited capacity losses which could not be explained solely on the basis of the observed active area reduction.

The percentage area reduction was calculated on the basis of X-ray examination of the Zn electrodes. Figure 1 shows an X-ray of the Zn electrode in cell PWR4 (10/90,40), both before and after cycling.

Figure 2 shows the capacity of cells PWR1 (const I,4), PWR3 (30/90,15), and PWR4 (10/90,40) plotted versus number of cycles. The capacities are given as a percentage of the design capacity (calculated as one-third of the ZnO material in the Zn electrode). The gaps in the data indicate cycles that were not representative of the behavior of the cells: e.g.,

1. Due to a software malfunction, cell PWR1 (const I,4) went through two very short cycles (49,50).

2. Cell PWR4 (10/90,40) was connected in series with cell PWR2 (10/90,40), which started failing at cycle 21. When cycled on its own, cell PWR4 returned to a higher capacity, comparable to that prior to cycle 21.

Peaks in the data are indicative of long discharges implemented to re-establish the ZnO content of the negative electrode (see Section 3.3.3).

Cell PWR3 (30/90,15) cycled at a constant capacity for most of its 107 cycles. The initial capacity decline was halted by gradual adjustment of the overcharge, up to 7%. Cell PWR4 (10/90,40) began to lose capacity early in life, and re-establishment of the ZnO reserve at cycle 56 did little to arrest the decline. Cell PWR1 (const I,4) stayed above 75% of design capacity for about 90 cycles. The ZnO reserve was recovered twice, and after the second recovery (cycle 75) the capacity declined rapidly.

T A B L E 1

Zn/NiOOH CELL CYCLING RESULTS

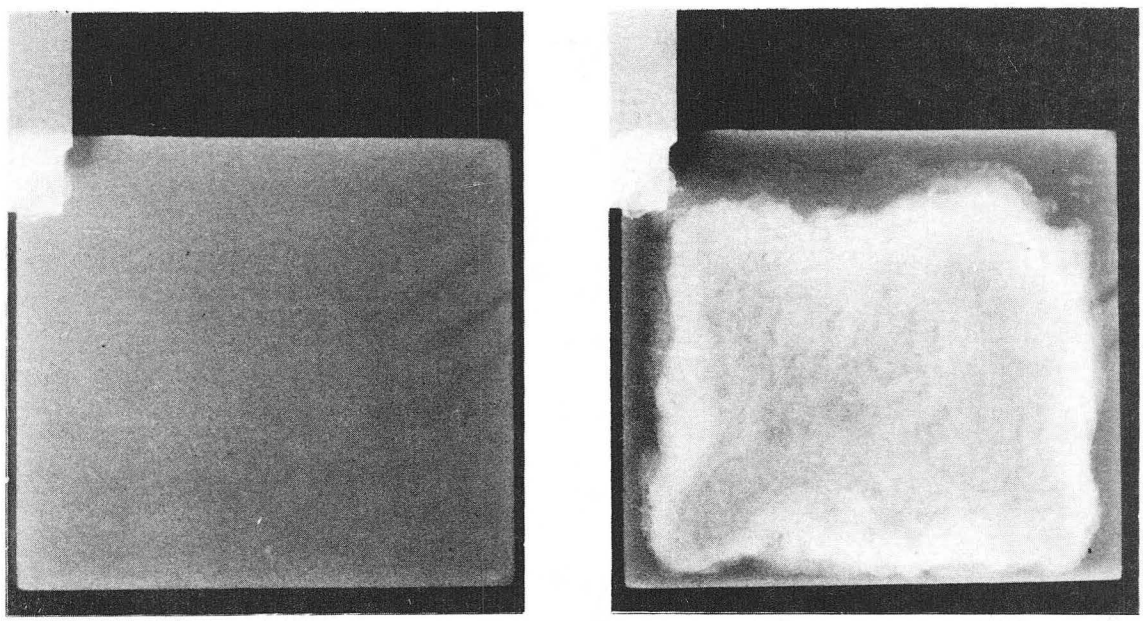
Cell	PWR1	PWR5	PWR3	PWR8	PWR6	PWR7	PWR2	PWR4
CHARGE MODE (ms ON/ms OFF)	CONSTANT CURRENT	8.3/8.3	30/90	30/90	30/90	30/90	10/90	10/90
CHARGE TIME (Hours)	4.5	4.5	4.5	4.5	2.25	2.25	4.5	4.5
CHARGE PEAK CURRENT DENSITY (mA/cm^2)	4.1	8.4	15.5	15.5	30.1	30.1	40.6	40.6
CHARGE PER PULSE (A-hr x 10^{-6})	---	1.7	11.2	11.2	21.7	21.7	9.7	9.7
CYCLES ABOVE 75% CAPACITY**	90 N	50 N	>107 N	*	43 N	31 Z	20 Z	34 Z
% AREA REDUCTION (Cycles)	25 (100)	19 (107)	18.5 (107)	*	30 (76)	32 (61)	12 (25)	31 (88)
% AREA LOSS PER CYCLE	.25	.18	.17	*	.39	.52	.48	.35
% CAPACITY LOSS (Cycles)	49 (100)	32 (107)	22 (107)	*	53 (75)	61 (61)	68 (25)	68 (88)
% CAPACITY LOSS PER CYCLE	.49	.30	.21	*	.71	1.00	2.72	.78

Notes:

* PWR8 incomplete -- still under test

** N = nickel - electrode limiting at end of test

Z = zinc - electrode limiting at end of test



XBB 826-5188

(a) (b)

FIGURE 1. X-ray Picture of a Zn Electrode (cell PWR4)

(a) Before cycling

(b) After cycling

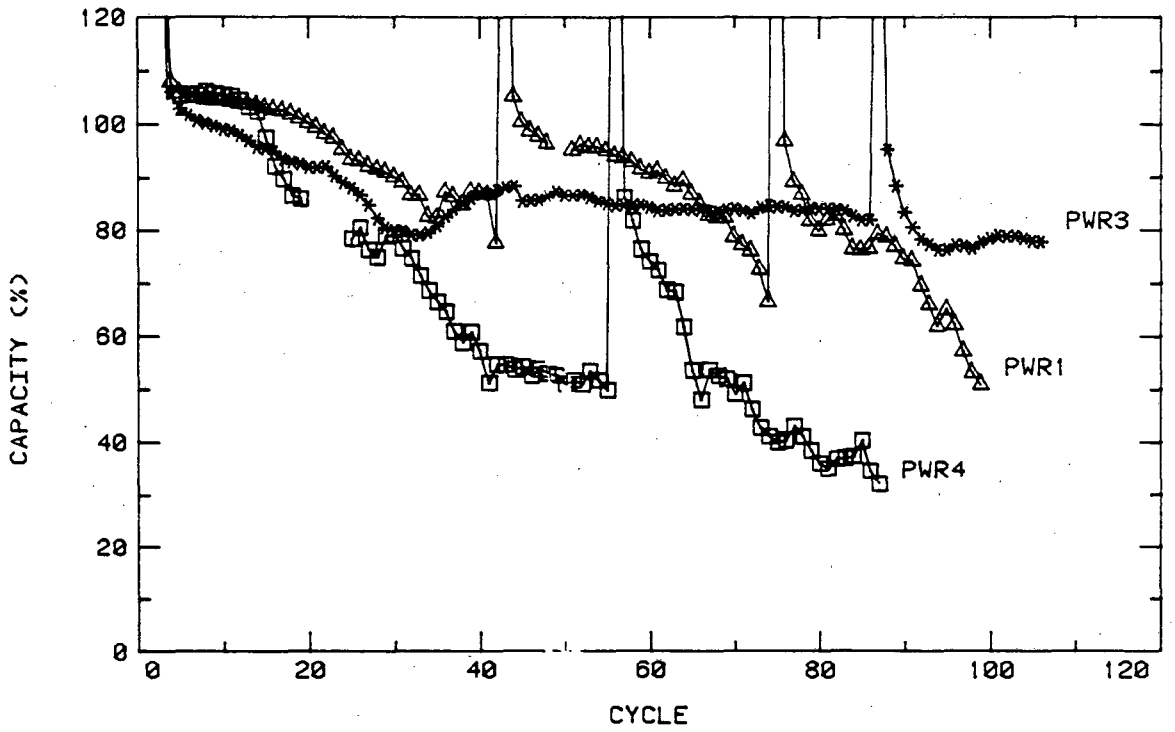


FIGURE 2. All identification and cycling conditions are given in Table 1. The cells are discharged to a 1V cutoff. Peaks in the data indicate discharges to re-establish the ZnO content of the negative electrode. Cells PWR1 (const I, 4), PWR3 (30/90, 15) and PWR4 (10/90, 40) are compared.

Figure 3 shows the capacity of cells PWR5 (8/8,8) and PWR7 (30/90,30) versus cycle number. Cell PWR5 held its capacity above 75% of the design value for about 50 cycles, but after a second ZnO reserve recovery (cycle 57), it displayed very dramatic decline. At cycle 68, the overcharge was increased to 20%, and then retained at 10% for the next 25 cycles: the cell did regain some capacity, but did not reach the original level.

4.2 COMPARISON OF CELLS PWR1 (const I,4), PWR3 (30/90,15) AND PWR4 (10/90,40)

Figures 4 and 5 show cell voltages for the three cells during charge, at cycles 30 and 69. The charge profiles are typical, and it appears that the cells behaved quite differently from each other.

It was straightforward to calculate the "resistance" of the cells which were charged with pulsed current (PWR3 and PWR4), since the program measured the cell voltage immediately prior to current interruption and within 100 msec after current interruption. No comparable measurements were available for cell PWR1 (const I,4). At cycle 30, the resistance of cell PWR3 (30/90,15) was 12 mΩ, and that of cell PWR4 (10/90,40) was 28 mΩ (both measured at the end of charge). At cycle 69, the values were 28 and 45 mΩ respectively. Note that these measurements include activation overpotentials and ohmic losses, but mass transfer overpotentials should be excluded, because they relax more slowly. Plausible explanations for voltage differences between the charge curves for PWR3 and PWR4 are

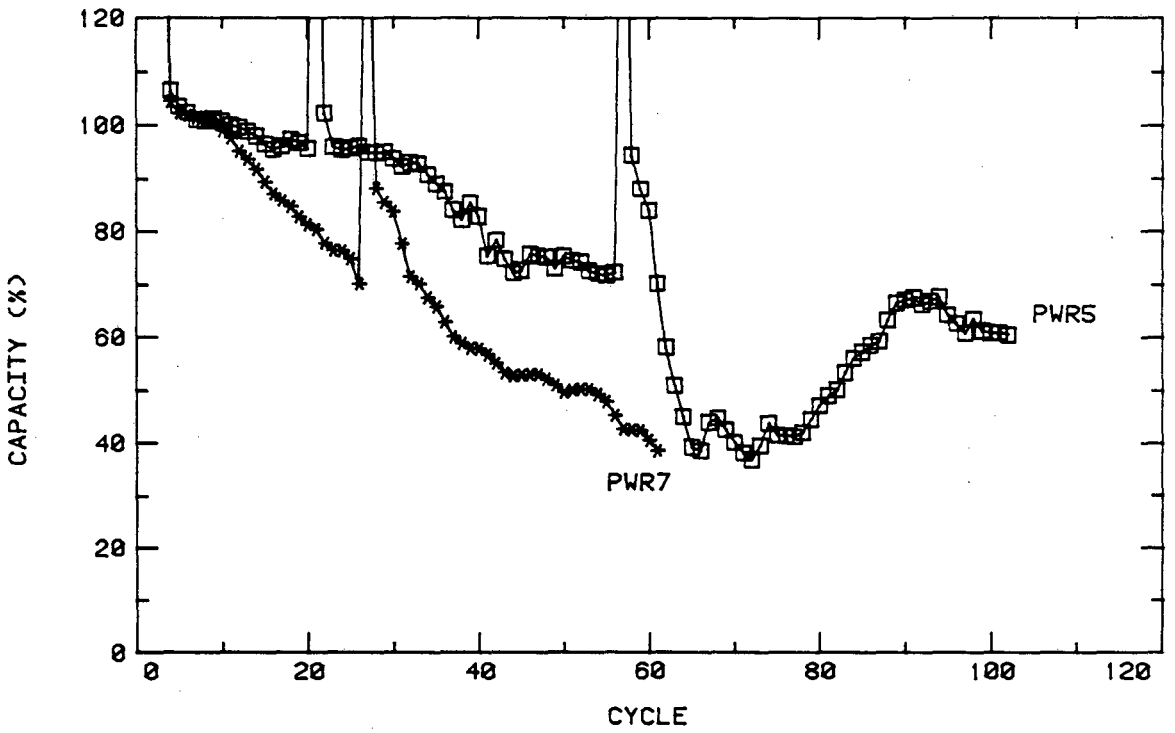


FIGURE 3. CAPACITY FOR CELLS PWR5 AND PWR7

XBL 826-803

Specifications as in Figure 2.
Cells PWR5 (8/8, 8) and PWR7
(30/90, 30) are compared.

1) higher ohmic drop due to higher peak current density in PWR4.

2) higher ohmic resistance (due, for example, to gas bubbles evolved)

3) higher activation overpotentials.

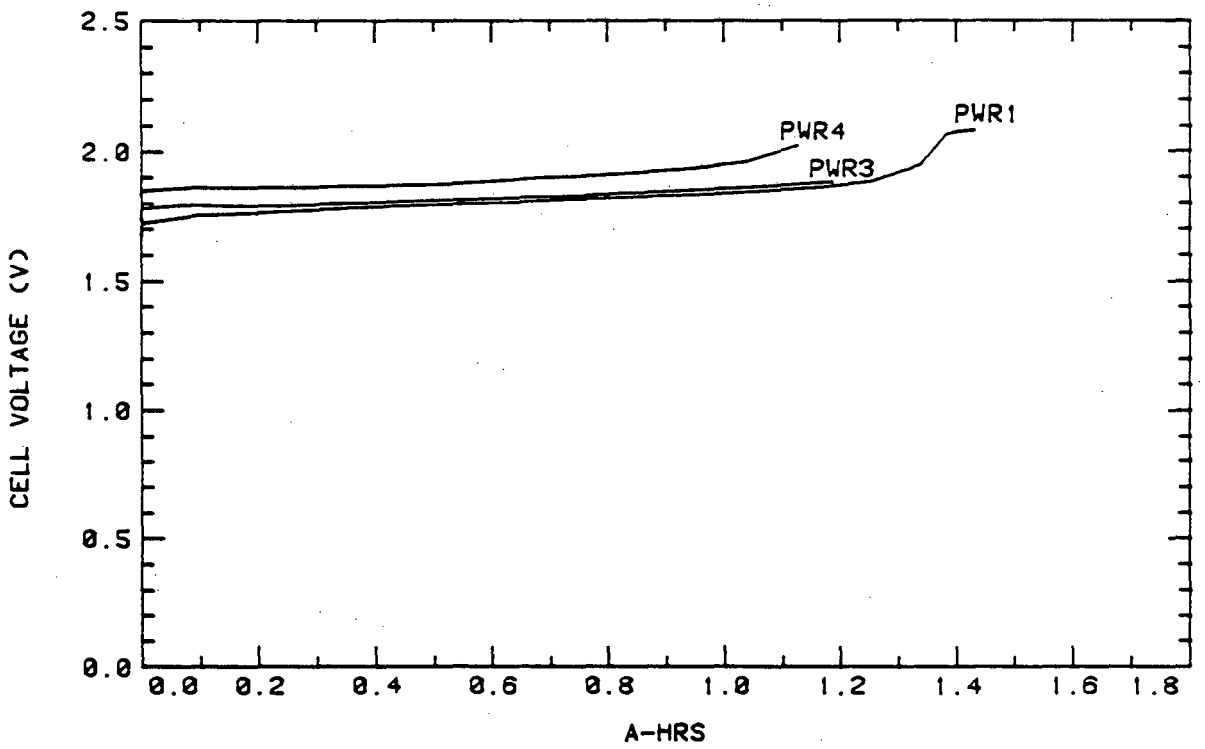


FIGURE 4. Effect of Various Charging Techniques on Charge Cell Voltage.

All identification and cycling conditions are given in Table 1.

PWR3 (30/90, 15) and PWR4 (10/90, 40) are compared.

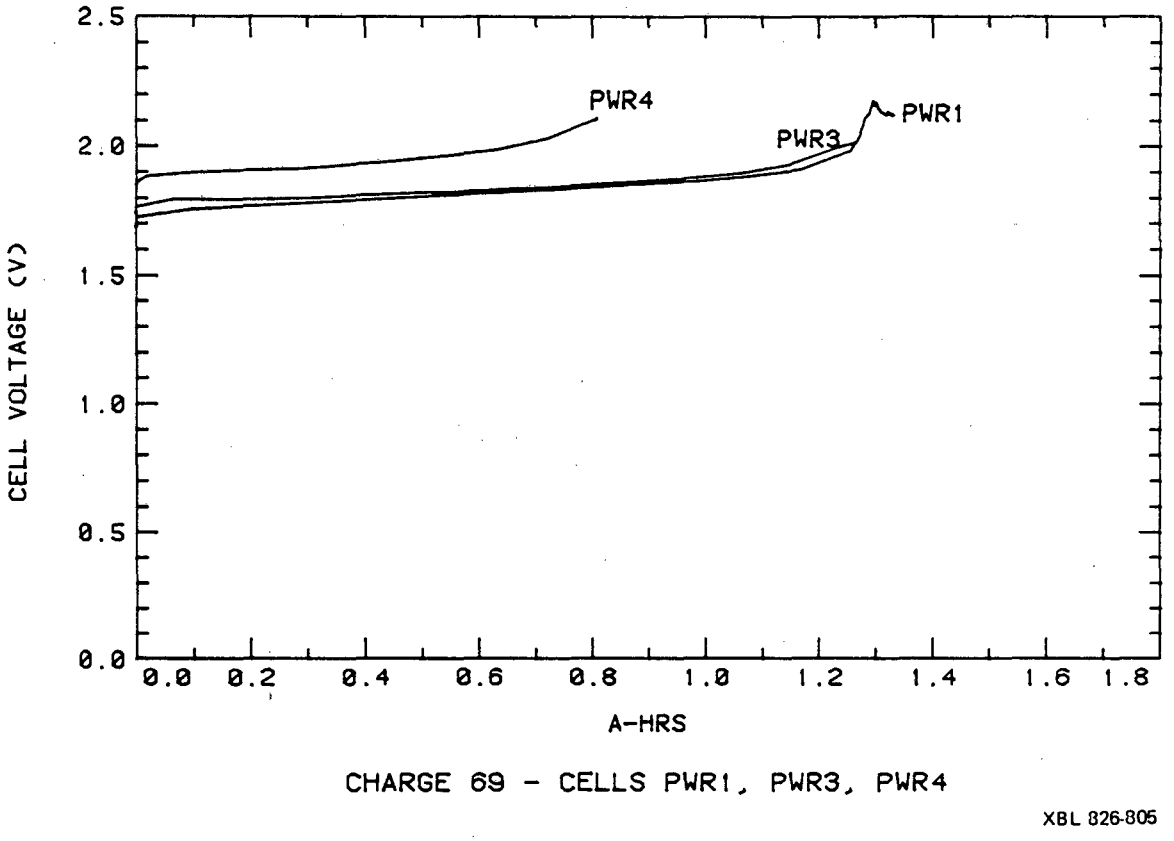


FIGURE 5. Effect of Various Charging Techniques on Charge Cell Voltage. Designations as in Figure 4.

4.2.1 CELL PWR1 (Constant Current, 4 mA/cm^2)

This cell was cycled with a constant-current charge. At cycle 20, the potential difference between the reference and Zn electrode (measured at open circuit after charge) was about 1.4V, which typically indicates a shortage of reducible ZnO material at the electrode. This condition gradually led to Zn dendrite growth, albeit slowed by the separator, until shorting occurred. At that point, the capacity showed a sharp decline, and even after re-establishment of the ZnO reserve a similar behavior pattern was observed.

Figure 6 shows charge-discharge voltage plots for cell PWR1 at cycles 10 and 73. The occurrence of a short at cycle 73 is obvious. Examination of the separator and wicks (after termination of the test) provided additional evidence that shorts had occurred. Several black (Zn) spots were visible on the wicks, and the inner layers of the separator appeared to contain some Zn material. In addition, the nickelized layer of the separator was partly separated from its Celgard support.

4.2.2 CELL PWR3 (30/90,15)

For this cell, the Zn versus reference potential (measured at open circuit after charge) reached -1.4 V after 62 cycles, much later than observed for the cell cycled with constant current (Section 4.2.1). Two explanations are proposed:

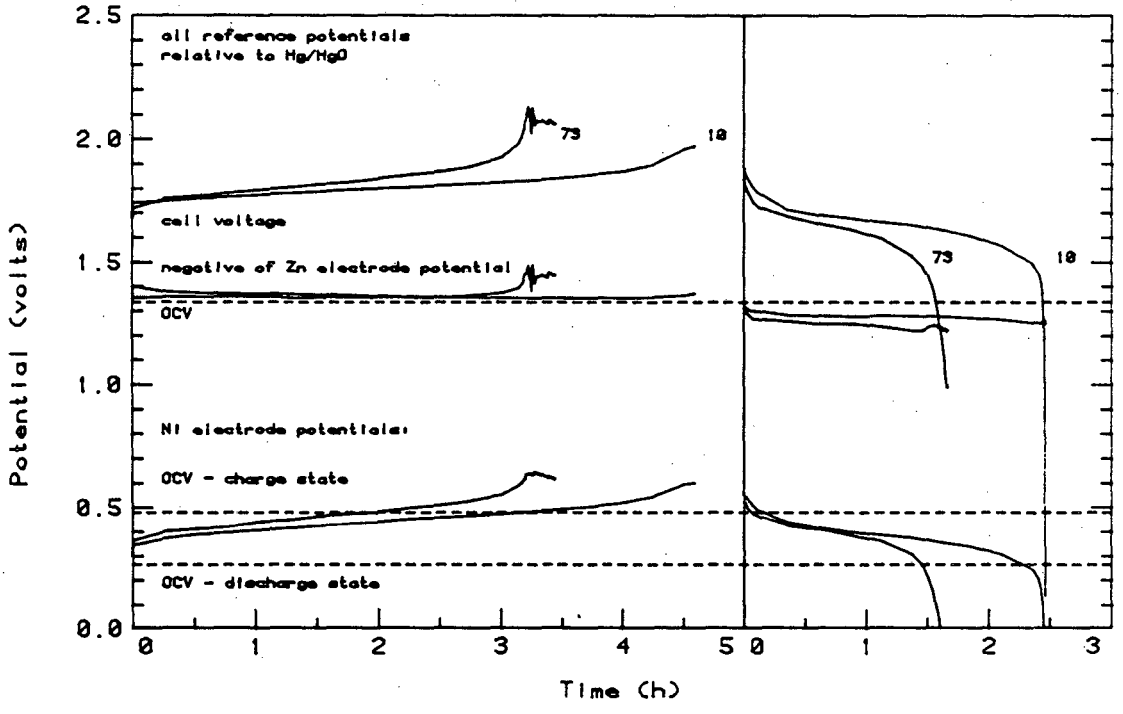


FIGURE 6. CELL VOLTAGE AND ELECTRODE POTENTIALS, CYCLES 10 AND 73, CELL PWR1

XBL 826-806

Charge: Left portion of graph

Discharge: Right portion of graph

Constant current, 4 mA/cm^2.

1) The ZnO reserve required more time to diminish because of an increase in hydrogen evolution rates during charge (the current density was equal to four times that employed at constant current). This phenomenon would prevent the Zn electrode from operating at 100% efficiency.

2) Even with a small ZnO reserve and a 15 mA/cm^2 current density, a 90 msec rest time after each pulse allowed relaxation of concentration gradients and improved mass transfer characteristics at the Zn electrode.

On discharge, this cell was never limited by its Zn electrode. Its separator was in good condition at the end of cycling; no Zn was detectable in the inner layers, and no traces of shorts were visible.

4.2.3 CELL PWR4 (10/90,40)

The performance of cell PWR4 was inferior to that of cells PWR1 (const I,4) and PWR3 (30/90,15). Its Zn versus reference potential (measured at open circuit after charge) reached -1.4V after only 10 cycles. Between cycles 10 and 20, the cell lost approximately 20% of its capacity, and the charge-discharge curves indicated that this decline might be related to cell shorting. The occurrence of shorts was corroborated by post-test examination of the separator and the wicks.

Cycles 20 to 39 showed low overpotential on the Zn-electrode on charge. After cycle 35, the Zn electrode became capacity-limiting on discharge. After cycle 40, this limitation was coupled with a rise in Zn overpotential on charge. Figure 7 shows two cycles for this cell: cycle 15 (high overpotential on the Zn electrode on charge, with

shorting toward the end of charge), and cycle 40 (high overpotential on both charge and discharge). For the charge portion of the graph, the dot-dash curves represent the cell voltage at current ON; the solid curves correspond to current OFF. As mentioned in Section 3.3.5, no valid reference electrode measurements could be obtained when pulsed charging was employed.

It is not unexpected that a cell which is charged at high current density can develop problems at its Zn electrode even quite early in life. Shorting was clearly one problem, but other mechanisms could have caused the Zn electrode to become the capacity-limiting electrode to a larger extent than what might be calculated from the reduction of active area. Some possibilities are:

1) A larger amount of hydrogen evolution resulted in loss of metallic Zn reserve.

2) Dendrite growth stoppage either by the separator or by current interruption could lead to isolation of Zn metal from the current collector.

The separator was in very poor condition; the nickelized layer had peeled off from its support, and the inner layers appeared to be heavily loaded with Zn material. This last observation provides support for the second mechanism of capacity loss.

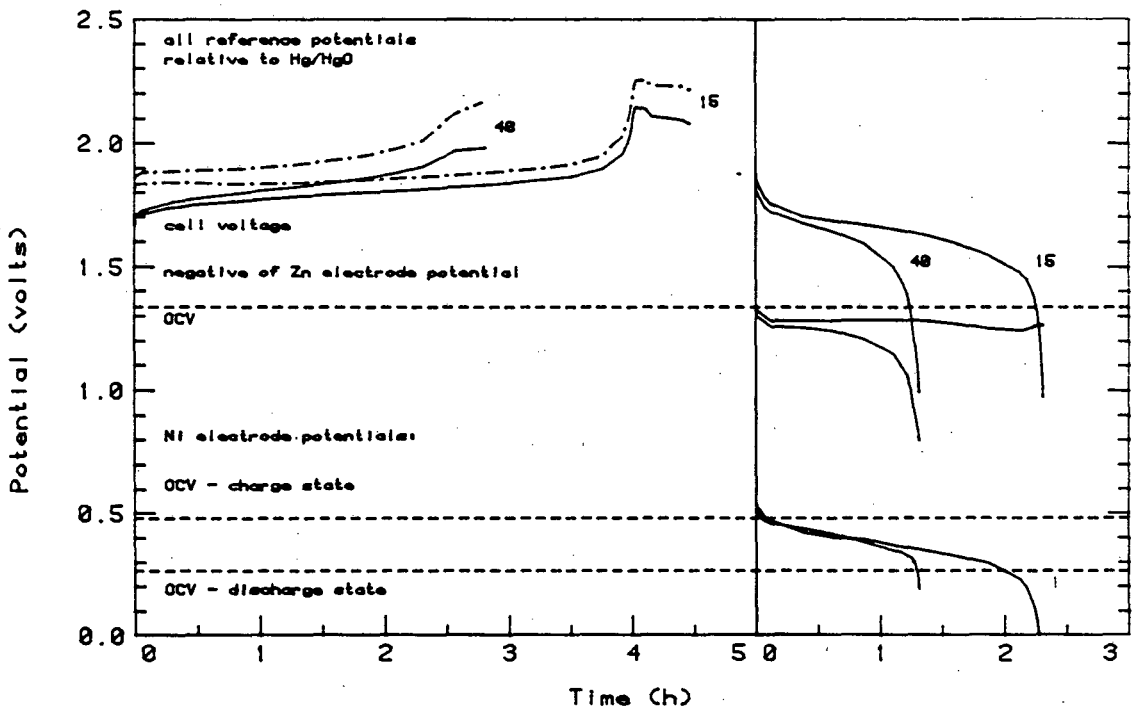


FIGURE 7. CELL VOLTAGE AND ELECTRODE POTENTIALS, CYCLES 15 & 40, CELL PWR4

XBL 826-807

Charge: Left portion of graph
Dotted lines: Measurements at current on
Full lines: Measurements at current off
No reference electrode measurements

Discharge: Right portion of graph

Cell PWR4 (10/90, 40) is examined.

4.3 CELLS PWR2 and PWR4 (10ms-on/90ms-off, 40 mA/cm^2)

Cell PWR2 was cycled in series with PWR4, and for 20 cycles it displayed the same electrical behavior, but it subsequently lost capacity very rapidly and became limited by its Zn electrode. At cycle 25, experimentation was terminated. Characteristics of its capacity decline and failure resembled that of cell PWR4, albeit at an accelerated rate.

4.4 COMPARISON BETWEEN CELLS PWR1 (const I,4) AND PWR5 (8/8,8)

Cell PWR5 was charged with a 60 Hz square pulse of 8.4 mA/cm^2 peak current density (the off/on ratio of this pulse profile is 1, and the off-time is 8.3 msec). On the basis of evaluations reported in the literature[1] no major improvements over constant current (cell PWR1) were expected for this charge method. It is, however, one of the easiest charge methods to implement in practice and therefore warrants study.

The Zn versus reference electrode potential (measured at open circuit after charge), reached a value of -1.4V at cycle 8 and continued at <-1.4V for more than 50 cycles. This can be explained by a current density equal to twice that imposed at constant current, and an off-time insufficient to compensate for the high current via improved mass-transfer conditions.

Figure 8 shows typical voltage profiles for this cell. The charging curves indicate a condition of gas evolution and eventual shorting. The cell was thus operating at low efficiency.

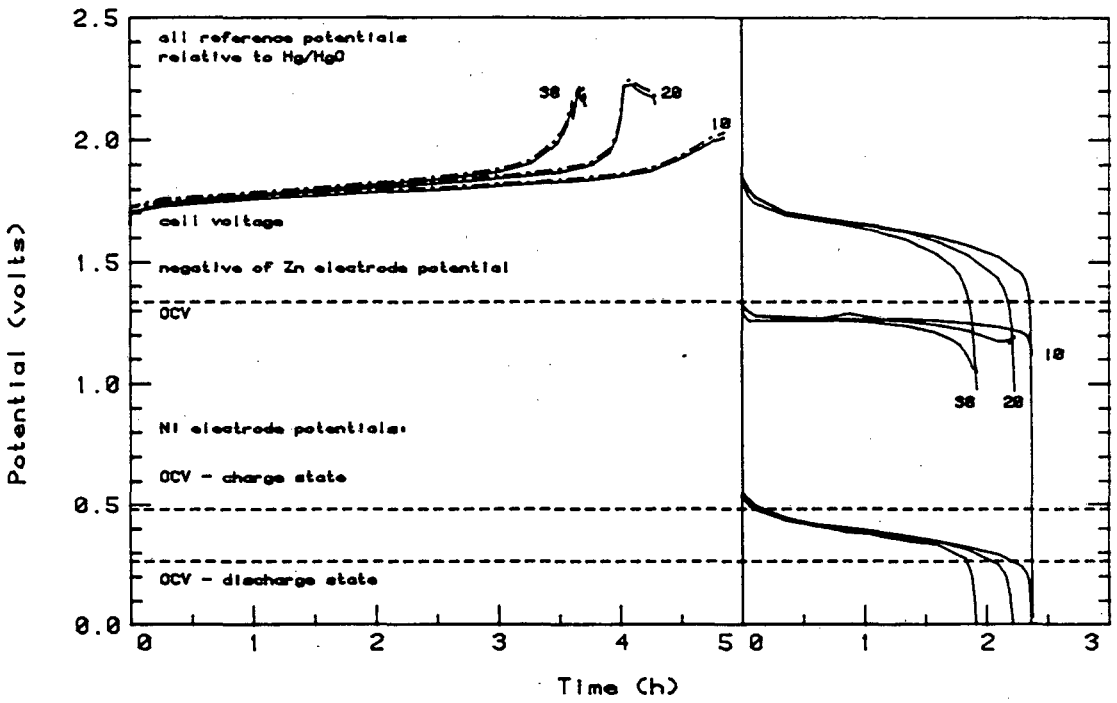


FIGURE 8. CELL VOLTAGE AND ELECTRODE POTENTIALS, CYCLES 10, 20, 38, CELL PWR5

XBL 826-808

Specifications as in Figure 7.

Cell PWR5 (8/8, 8)

After a second restoration of ZnO reserve capacity, the total capacity dropped from over 70% to about 35% in 10 cycles. This loss was not associated with problems at the Zn electrode, and heavy overcharges were required to gradually restore the capacity to 65%, at which point the capacity again declined, and the cell operation was terminated.

4.5 CELLS PWR6 (30/90,30) AND PWR7 (30/90,30)

Experiments on these two cells were intended to investigate charging regimes intermediate between cell PWR3 (30/90, 15.5 mA/cm^2) and cells PWR2 and PWR4 (10/90, 40.6 mA/cm^2). The profile was the same as that employed for PWR3 (30/90,15), but with twice the current density. The two cells showed poor reproducibility in their cycle life.

Cell PWR7 behaved in a manner quite similar to that of cell PWR4 (10/90,40), and its rate of capacity decline was sightly higher. As was the case for cells PWR2 (10/90,40) and PWR4 (10/90,40), cell PWR7 (30/90,30) rapidly became limited on discharge due to a high overpotential at the Zn electrode. Figure 9 shows the charge curves of cells PWR4 and PWR7 at cycle 26. Higher voltages exhibited by PWR7 can be explained by the fact that PWR7 passed more than twice the number of coulombs at current-ON (Table 1), for the same 90 msec current-OFF time.

Cell PWR6 (30/90,30) displayed peculiar behavior: its Zn electrode was never limiting (neither on charge nor on discharge), and it is suggested that the NiOOH electrode exhibited anomalously low efficiency.

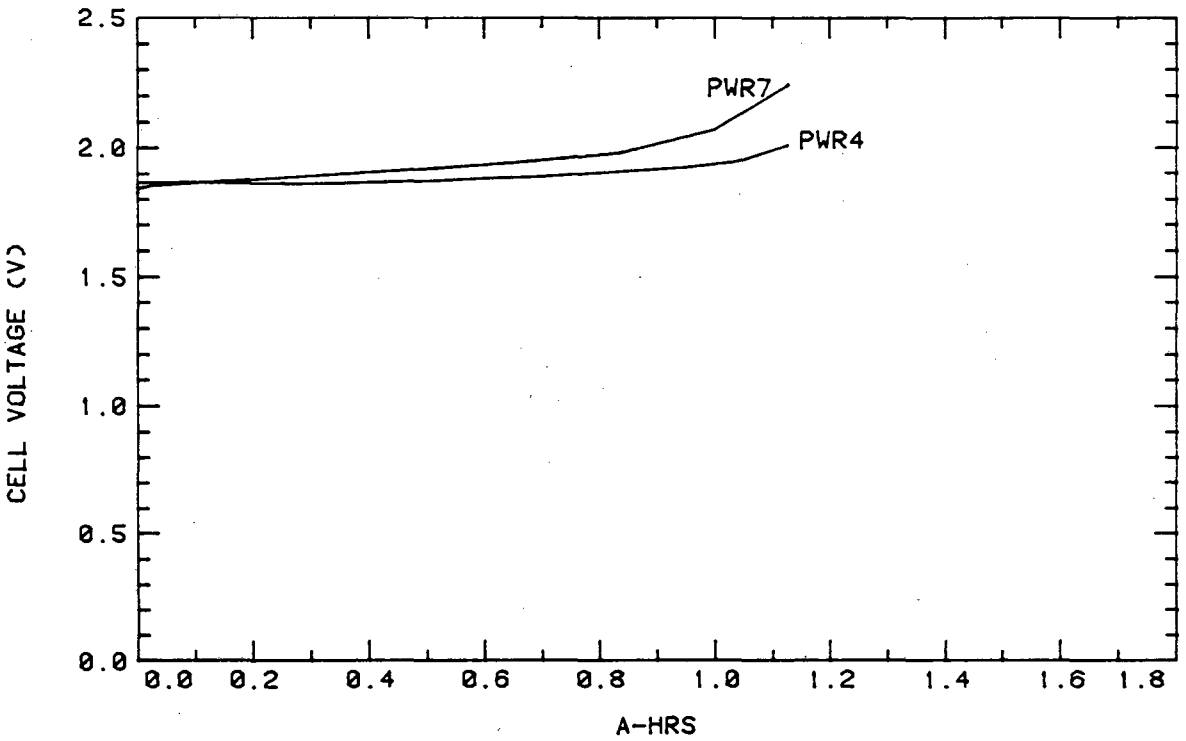


FIGURE 9. Effect of Various Charging Techniques on Charge Cell Voltage

Comparison of cells PWR4 (10/90, 40) and PWR7 (30/90, 15)

4.6 SUMMARY OF RESULTS

Various mechanisms can cause a cell to lose capacity. Some of these are interrelated: shape change (I), densification (II), passivation (III), dendrite shorting (IV), inefficiency of the Zn electrode (H_2 evolution) (V), inefficiency of the NiOOH electrode (O_2 evolution) (VI), entrapment of gas bubbles in the electrode pack (VII), loss of active Zn material due to disconnected dendrites (VIII), isolated Zn on the electrodes (IX), Zn material in the NiOOH electrode (X), etc. Because of the cumulative effects of these interrelated mechanisms throughout the cycle life of a cell, it is difficult to isolate the contribution of a single mechanism to capacity loss and/or final failure. This shortcoming might be overcome by additional data and more extensive ancillary experiments. Some relevant data might have been gathered from quantitative analysis of the gases evolved during charge; precise mass balances for the electrodes, the electrolyte and the separator; determination of the amount of undischarged Zn in the electrode, SEM analysis, etc. Moreover, the cycling tests performed demonstrated marginal reproducibility in some cases. Though the analysis of accumulated observations and data has not unequivocally identified major failure mechanisms or unambiguously associated capacity decline rates with particular physico-chemical phenomena, it has suggested charging modes which offer the promise of improved capacity retention of Zn/NiOOH cells.

One conclusion that can be drawn from the experiments is that shape change did not play a major role in cell failure. The only cell that did not fail in less than 100 cycles (cell PWR3 - 30/90,15) was also the only one where shape change rates and capacity decline rates correlated with each other (see Table 1). The remaining cells exhibited rates of shape change more rapid than that of cell PWR3, but rates of capacity

decline were greater still.

Additional information can be derived from analysis of the voltage curves, and from the fact that some cells were limited on discharge by their Zn electrodes, others by their NiOOH electrodes. The latter case is more likely caused by mechanisms IV, VI, VII, and X, while the former would be the result of the other mechanisms. Shape change (I) was already ruled out as a determining factor. The Zn electrodes were sufficiently flexible after test to discard densification (II) effects and, perhaps, passivation (III) effects, since these are often related. For every cell, the NiOOH electrodes were weighed after test, and they were invariably heavier after than before the test, indicating the possible presence of Zn material in the NiOOH electrodes (X). Since the weight gains were roughly equal for all Ni electrodes this effect cannot differentiate one cell from another.

The cell cycled at constant current (PWR1) probably lost most of its capacity through shorting. This is partly due to the fact that the ZnO reserve was not restored as soon as the overpotential on the Zn electrode began to rise. An important observation was that the separator was not resistant enough to prevent eventual dendrite penetration. The cell cycled with square pulse (PWR5 - 8/8,8) at 60 Hz also suffered from shorting, although less pronounced.

Cells PWR2 (10/90,40), PWR4 (10/90,40), and PWR7 (30/90,30) were all limited by their Zn electrode on discharge. All combined a high current density with a relatively large number of coulombs passed per pulse. This regime could have favored mechanisms V and VIII (evidence for the latter case is provided by the presence of Zn material in the separator). Aside from a short period in the initial portion of the test on PWR2 and PWR4, the voltage curves did not reveal signs of heavy shorts.

Finally, cell PWR3 (30/90,15) seemed to combine all possible advantages: it was cycled with a 30ms-on/90ms-off charge pulse (which was good for mass-transfer conditions) and a moderate current density. This cell did not show evidence of shorting or excessive overpotential on either electrode, and its ZnO reserve required reestablishment only late in the experiment. There were no signs of shorts on the wicks, no Zn material in the separator, and the voltage curves were very uniform during the whole experiment. It can therefore be said that this is a promising charging mode and is certainly worth further investigation.

REFERENCE

(1) O. C. Wagner, "High Cycle Life, High Energy Density Nickel/Zinc Batteries", Report No. 3, U.S. Army Electronics Research and Development Command, Fort Mammoth, New Jersey, Report No. PSD-3C, February 1980.

CONCLUSIONS

A multiple electrochemical cell test system has been designed and constructed. It has been demonstrated to have the capability to simultaneously and continuously cycle 16 electrochemical cells under a variety of test regimes.

A number of Zn/NiOOH cells constructed in this laboratory have been cycled with various charging modes in order to investigate their possible effect on cycle life and capacity retention. The cells were cycled at 100% depth-of-discharge, and the charging modes compared were constant and pulsed currents, with a variation in duty cycle and amplitude.

The best charge method was found to be a 30ms-ON/90ms-OFF charge pulse, with a 15mA/cm^2 peak current density (4.5 hour charge time). Two cells were charged with this mode. Of these two, one showed significantly better capacity retention than any of the other cells cycled under differing modes, and the second cells is still under test. Application of the other testing modes resulted in the following behavior:

1) The rate of capacity decline was higher than the rate of area reduction, indicating the presence of failure mechanisms other than shape change.

2) High overpotentials on the Zn and/or NiOOH electrodes on charge, and/or on the Zn electrode on discharge developed early in the cells' life.

The two cells cycled with the 30ms-ON/90ms-OFF, 15mA/cm^2 pulse did not exhibit high overpotentials during their cycle life. Thus, this pulse provided favorable cell cycling conditions and considerably slowed the rate of failure and capacity-decline mechanisms observed for the other cells.

ACKNOWLEDGEMENTS

My greatest thanks go to Dr. Elton Cairns and Dr. Frank McLarnon for their patient guidance; to my parents and friends for their love and support; to Randy Michelson and Joe Katz for their technical advice; and to Kathy Ellington, Eva Edwards, and Valerie Kelly for their kind and friendly help.

Financial support by the Assistant Secretary for Conservation and Renewable Energy, Office of Energy Systems Research, Energy Storage Division of the U.S. Department of Energy under contract No. DE-AC03-76SF00098 is gratefully acknowledged.

APPENDIX A

HARDWARE SYSTEM

A.1 SYSTEM DESCRIPTION

A block diagram of the multiple electrochemical cell test system is shown in Figure 1. The system presently employs 8 current controllers (n=8) and can accomodate up to 16 cells under test at any one time. The number of controllers may be easily expanded, as illustrated in Figure 1. The current controller is a set-point type and maintains the current by means of an analog feedback control loop at a value proportional to the analog control voltage input.

A.1.1 CURRENT CONTROLLER

The current controller provides for selection of either computer or local (manual) control of both charge and discharge currents. Control means the current is maintained at the set point level independent of subsequent changes in the cell's internal impedance, variations in the external power supply, or current changes in any of the other cells. Local control is used for setup, test or manual operation. According to the type of test, control of current may require the current to follow a constant power or a particular power or current demand. The controller must have sufficient bandwidth to deliver or sink a current such as demanded by the EPA power profile (see Figure 2, Appendix G). Pulsed current waveforms place even more stringent demands on the current controller response time. The current controller time constant of 1 ms provides a transient response adequate to supply the rapidly fluctuating currents required for many electrochemical experiments.

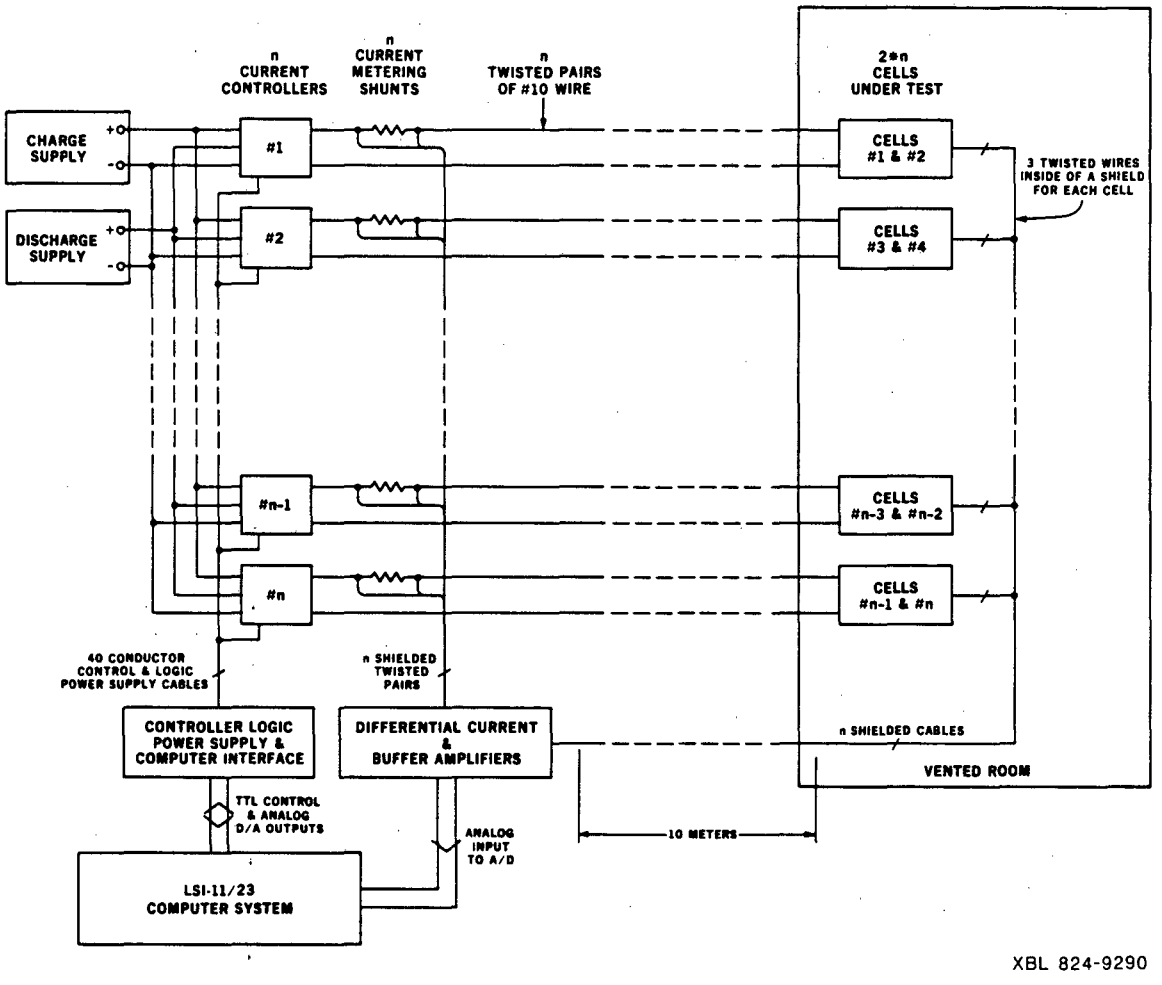


FIGURE 1. Multiple Electrochemical Cell Test System Block Diagram.

The interconnection block diagram is illustrated for n independent current controllers. Cabling is detailed for each of n controllers with two cells connected in series under test.

The task of monitoring both cell current and cell voltages is performed by the computer system. A decision to terminate a test or to change the direction of current flow is made by software logic. There are no voltage comparators and hardware logic to control current based on cell voltages. The current controller is designed to perform the task of analog current control on a continuous basis. The setpoint control voltage is usually supplied by the microcomputer system, or it may be supplied by an external function generator, for example, when rapidly pulsed currents are desired.

Functional control of the current controller is by front panel switches or 0 to 4V logic level control lines (TTL) from the computer system. The computer system must provide three output control lines for each current controller. One output line determines the direction of the current flow (charge or discharge). A second output control line is used to connect or disconnect the cells under test by means of a relay. This provides the ability to completely isolate any cell from any current source or load. The third output control line specifies the set point control source: an external function generator, or the computer's digital to analog converter. Additionally, an input line from each controller to the computer is required to monitor the state of the front panel switch that assigns computer or local (manual) control.

A.1.2 POWER DOWN AND SAFETY

The current controllers must provide for a safe shut-down under conditions of power failure or interruption. A sudden power failure must not cause hazardous conditions, damage to the cells (overcharge or overdischarge), or interrupt the test in such a manner that easy resumption

of operation is prevented. The controller digital logic is designed so that when powered up the relay is open, disconnecting the cells under test from the controller. In the event of a power failure the cells are disconnected and are not reconnected except by manual command (after power has been restored). The controller digital logic is designed so that a disconnect command will reduce the cell current to zero before the relay may open, which permits use of relatively low-current, inexpensive relays. If the computer control program should fail, the controller digital logic timer circuit will disconnect any cells under test within 2 minutes. The controller disconnect timer must be reset at least once every 2 minutes by a computer command on any logic level control line to any current controller.

A.1.3 LOGIC SUPPLY AND INTERFACE

The controller logic power supply and computer interface contains power supplies with sufficient capacity to energize the digital and analog logic in all of the current controllers. Each current controller requires 3 output control lines from and 1 input line to the computer. The computer also must provide an analog set point control voltage. The timeout logic protection is naturally located in the interface, and it is the only digital logic in this device. The interface is a flexible and easily-expandable means of distributing, to each controller, the input and output (I/O) control lines from standard computer I/O devices which are described more fully in Section IV below. In effect, the interface is a junction board for many mass-terminated 40 pin cables. Each controller, through the interface, is provided with logic power, control lines, the analog set point control voltage, and the timer-

controlled connect/disconnect enable line by means of wired busses and jumper wires on a printed circuit board. The power supplies have spare capacity and the wiring may be easily expanded to provide for more than 8 current controllers.

A.1.4 ANALOG RESOLUTION

Presently, many commercial 12 bit (1 part in 4096, 0.025% of full scale) digital to analog (D/A) voltage sources are available as LSI-11/23 peripheral devices[1]. Choice of this accuracy for the set point control voltage source implies that all of the other system controls and measurements should aim for this level of precision. Therefore, the current controllers must be stable and reproducible to 0.025% of full scale and any crosstalk due to changes of current in other cells must be less than 0.025% of full scale.

Cell voltages and metering shunt voltages are converted by differential type analog to digital (A/D) converters. The Differential Current and Buffer Amplifier Box (Figure 1) is used to connect the cells and metering shunts to the A/D converters. The A/D converters have 12-bit precision with a nominal minus 5 volt to plus 5 volt input range. A differential amplifier with a gain of 100 is used to convert the (50 mV full scale) metering shunt voltage to 5 V full scale. At this voltage level, noise induced by digital signals and clocks in the CPU and disk drive does not induce <u>significant</u> errors. Additionally, the interface box contains unity gain buffers with less than 10 pico-ampere input bias current. These low-current buffer amplifiers are needed for potential difference measurements between each of a cell's electrodes and its reference electrode. It is necessary to reduce the load on these low

area (high resistance) reference electrodes when the measurements are made with external equipment, such as a strip chart recorder.

A.2 CURRENT CONTROLLER DESCRIPTION

The current controller circuitry is constructed on a single printed circuit card and mounted along with the high-power-dissipation control elements in a size 4 NIM module (2). A block diagram of the current controller is shown in Figure 2. Inside views of the current controller NIM bin are shown in Figures 3 and 4. The front panel controls, the rear panel connectors, and the relay to disconnect from the cell under test are shown in Figure 5.

A.2.1 CONTROL AND METERING SHUNTS

Full-scale current of the current controller is determined by the rating of the control shunt. The control shunt is schematically shown in Figure 2 and may be seen mounted below the bottom of the controller printed circuit card in Figure 3. Control shunts(3) are available in standard 1, 2, 3, 5 and 10 ampere sizes and the current controllers have been tested with each of these shunts. Long- and short-term changes in current of less than 0.025% of full scale were measured for both charge and discharge currents. It was appropriate to provide another metering shunt external to the controller in series with the current lines to the cells under test. This current metering shunt provides a voltage which reverses sign when the current direction is changed from charge to discharge. This voltage is used by the computer system to measure current, direction, and to confirm that there is in fact an output current from the controller. We have calibrated these relatively

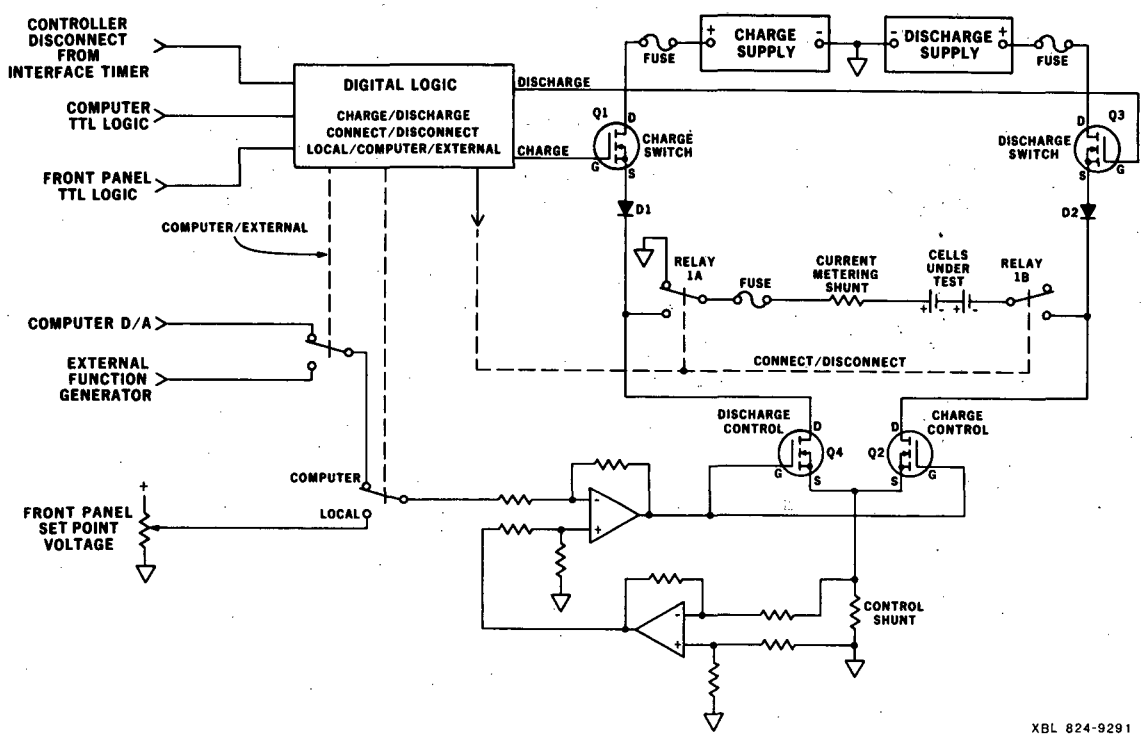
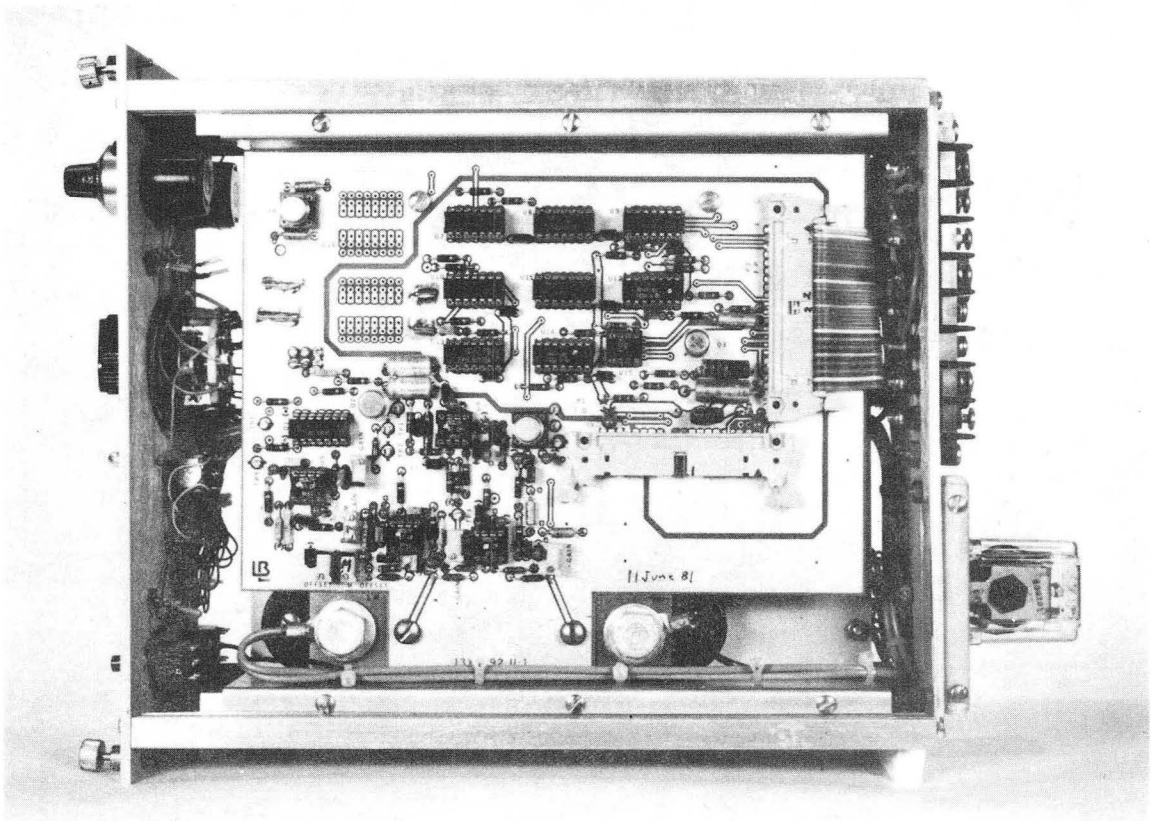


FIGURE 2. Current-Controller block diagram.

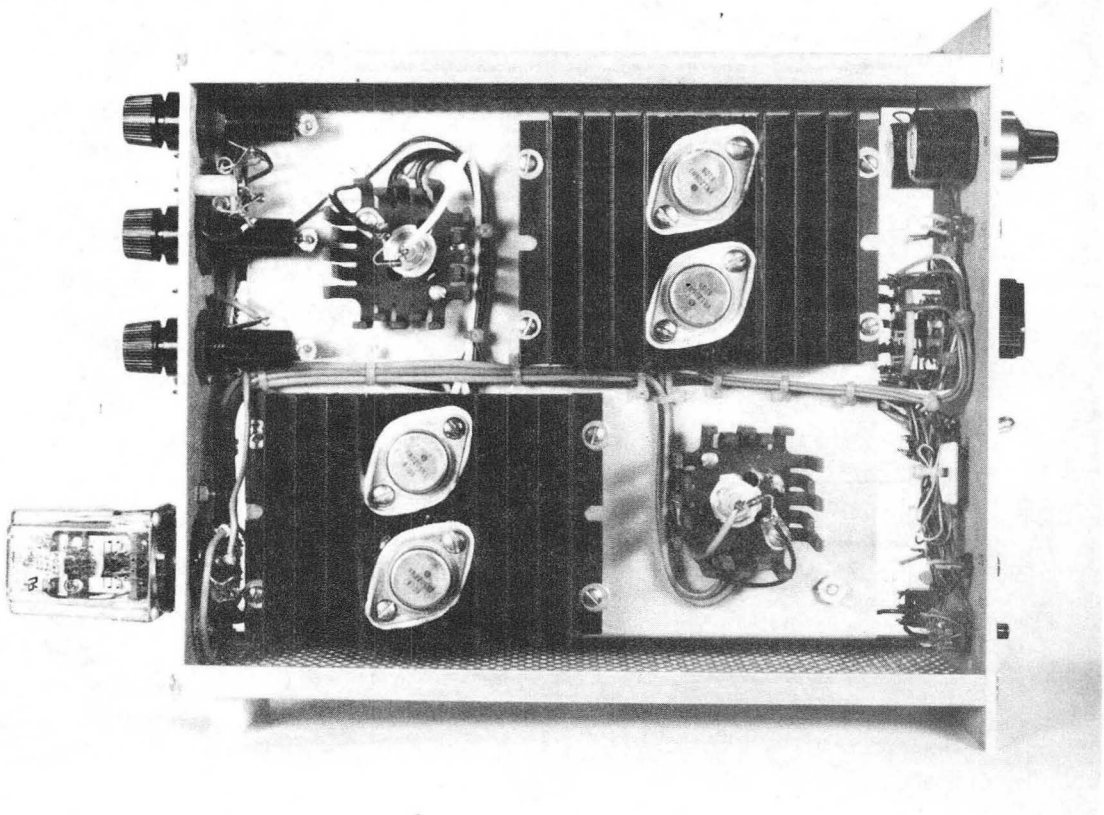
Current-controller digital and analog circuitry is shown with the relay positioned for the cells under test disconnected. When connected, relay contacts 1A and 1B will be closed. Charge current will then flow from the charge supply through Q1, D1, the current metering shunt, the cells under test, and the charge control (Q2), and it will return to the supply common via the control shunt. Discharge current will flow from the discharge supply through Q3, D2, the cells under test, the current metering shunt, and the discharge control (Q4), and it will return to the supply common via the control shunt.



XBB 822-1570

FIGURE 3. Inside View of the Current Controller, Logic Card and Control Current Shunt.

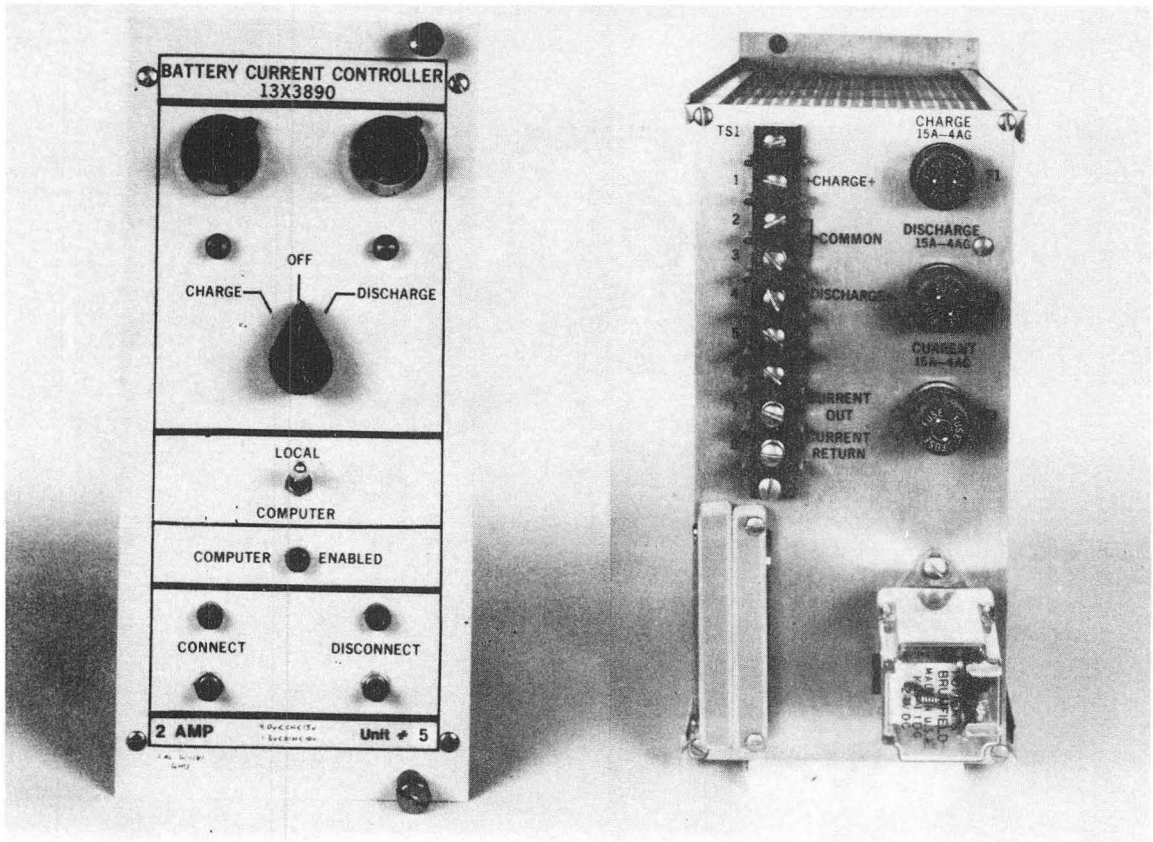
This side-view photograph of the current controller shows the printed-circuit logic card. The control shunt may be seen mounted directly on the bottom of the printed-circuit logic card.



XBB 822-1569A

FIGURE 4. Inside view of the Current Controller and Heat Dissipating Elements.

This photographic view highlights the heat dissipating elements located inside the current controller. Note that the charge VMOS switch and discharge control VMOS are mounted on one heat sink, while the discharge VMOS switch and the charge control VMOS are mounted on the other heat sink.



XBB 822-1568A

FIGURE 5. Current Controller, Front and Rear Panels.

Local (manual) front panel control of all functions is possible by means of the switches and push buttons shown. Indicators are provided to clearly display the present status of the current controller, even while under computer control. The fuses and relay are mounted on the rear panel for easy access. A 40-conductor flat ribbon cable passageway is provided at the lower left of the rear panel. By means of this cable the current controller is connected to the logic power supply and computer interface.

inexpensive shunts against National Bureau of Standards certified standards. Errors due to self heating are less than 0.025% of full scale except for the 10 ampere shunt at currents in excess of 8 amperes. Self heating errors are readily reduced by a small current of air blowing across the shunt.

A.2.2 VMOS CONTROL

The current controller analog circuit is a simple feedback control series regulator type. The linear dissipative element is a VMOS transistor (4) capable of dissipating 120 watts into an infinite heat sink. Figure 4 shows the arrangement of heat dissipating elements in the NIM module. Cooling air flow up through the NIM controllers is provided by an intake fan underneath (cool air intake from in front of the rack) and an exhaust fan above (exhaust air out to the rear of the rack). Cooling air flow is also directed across the control shunt, and the digital and analog control logic circuitry is shielded from the heat dissipating elements by a metal plate (Figures 3 and 4).

The VMOS transistor is a good linear control element and an efficient switch. By using two VMOS transistors, one as the linear control element and the other as a switch (Figure 2), one can readily switch the polarity of the cells under test. One may use a single power supply for both charge and discharge currents but power dissipation considerations for the linear control element illustrate the advantages of using separate charge and discharge power supplies. If we consider the supply voltage required to charge a cell, see Figure 2, we get:

$$V(\text{supply}) = V(\text{cell}) + V(\text{control VMOS}) + V(\text{drop}) \qquad (1).$$

where, V(cell) is the open circuit cell voltage, V(control VMOS) is the voltage drop across the control VMOS transistor biased in the linear region, and V(drop) is the sum of all the other voltage drops in the circuit. V(drop) is increased by the resistive losses in controller wiring and the cabling to the cells under test. Also included in the term V(drop) is the forward voltage drop in the steering diode, D1 (Figure 2). For minimum power dissipation in the control element we should keep the supply voltage to the minimum value necessary to insure that the control element has sufficient voltage to act as a linear control element. We can rewrite equation (1) as:

$$V(\text{control VMOS}) = V(\text{supply}) - V(\text{cell}) - V(\text{drop}) \qquad (2).$$

If we should now change the polarity of the cell under test, to discharge it at the same current as was used to charge it, we can write the voltage across the linear control element as:

$$V(\text{control VMOS}) = V(\text{supply}) + V(\text{cell}) - V(\text{drop}) \qquad (3).$$

V(drop) and the supply voltage will be the same as during charge. Therefore, initially the power dissipation in the control element will be larger by the product of the cell voltage times the discharge current. The increased power dissipation required in the control VMOS transistor will pose even a larger problem if one should try to test two cells in series. The use of separate charge and discharge supplies allows for the minimum amount of power dissipation in the control

elements and therefore less heat, resulting in longer life, greater accuracy and increased reliability for the current controller.

A.2.3 COMMON SUPPLIES

Both charge and discharge power supplies must have current ratings sufficient to supply the sum of the full scale currents for the current controllers. Both supplies may be inexpensive since load or line regulation is not required. The supplies must maintain their full load output voltage at a value greater than the minimum supply voltage required as shown in Equation 1. For example, with a 2 ampere full scale current controller it is possible to change the supply voltage from a minimum of 4 volts to a maximum of 15 volts and maintain constant current. (For this test the open circuit cell voltage was approximately 1.8 volts). At no time was the change in charge current due to a change in supply voltage greater than 0.025% of full scale. The excellent rejection of supply voltage change is also reflected in good current crosstalk performance.

A.2.4 CURRENT TIME RESPONSE AND CROSSTALK

Both the time response and crosstalk performance of the current controller may be seen in the photographs of oscilloscope traces shown in Figure 6. The top waveform is the external analog setpoint control voltage. The charge current response of a 10 ampere full scale current controller to the 7 volt, 10 millisecond pulse with a 100 millisecond period is shown by the middle trace. The 7 ampere change in charge current settles to within 0.01% of the final value within 5 milliseconds. The bottom trace shows typical current fluctuation caused by crosstalk to one of the remaining current controllers. At the time of the most rapid change in current, the common charge power supply voltage is changed. The current controller response is rapid enough, as may be

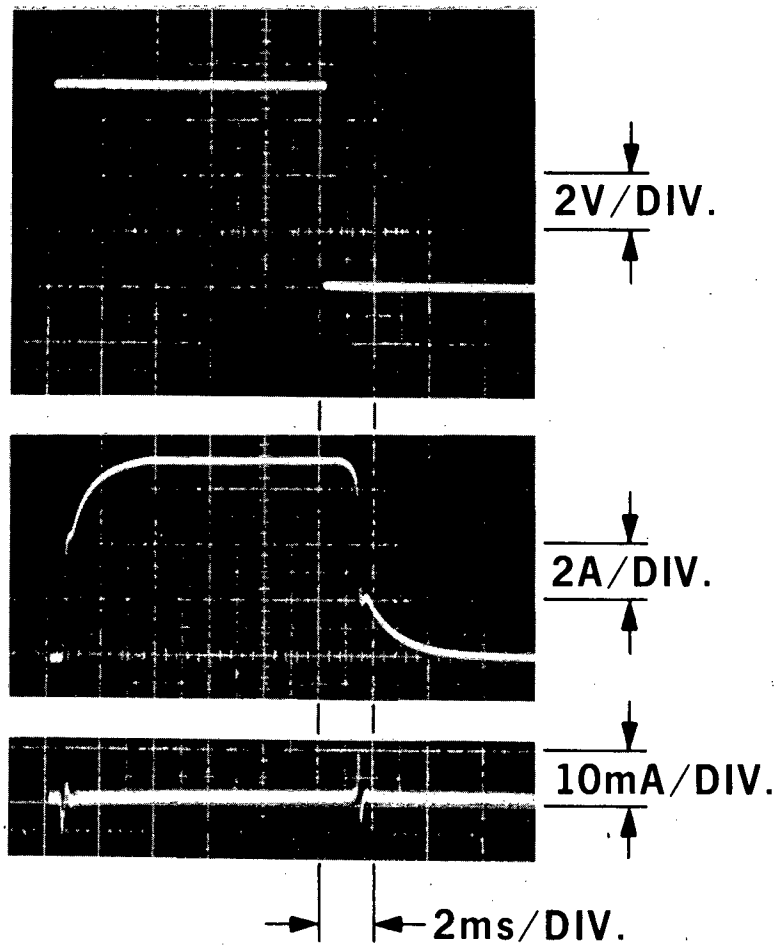


FIGURE 6. Current Response and Cross Talk Waveforms.

The upper waveform is a 7-volt (external function generator) set point control voltage. The middle trace is the corresponding 7-ampere charge current waveform. The lower trace is typical of the current change induced in any other current controller output due to the 7-ampere change shown by the middle trace.

seen in Figure 6, to limit the change in current due to crosstalk to less than 10 milliamperes, for a 7 ampere current change in another controller, with less than 0.4 milliseconds duration. The long term (greater than 0.4 milliseconds) change in current in any controller due to a full scale change of current in any other controller has been measured as less than 0.025% of full scale. Note the lower trace of Figure 6 shows no discernable shift in the steady-state value of the controller current.

A.2.5 PROTECTION AND EXPANSION

Each of the current controllers has a 15 volt protective diode on the charge supply input line and 12 volt protection on the discharge supply input line to prevent an accidental overvoltage condition or unnecessarily large power dissipation due to an external current supply failure or improper set up. Each of the supply input lines is fused for the full scale current as are the controller output lines to the cells under test.

The modular construction and the relative ease of paralleling VMOS transistors make it possible to extend the current range of these controllers to higher currents. One could readily drive many heat sink assemblies of 10A capacity (Figure 4) using appropriate control and metering shunts.

A.3 COMPUTER HARDWARE DESCRIPTION

A diagram of the LSI-11/23 (see Section 2.4) and its peripherals is shown in Figure 7; a description of these peripherals follows.

The LSI-11/23 chosen for this system uses memory with 16-bit words but has an 18-line addressing bus to allow the CPU to access 128K words of memory. The system is installed with 128K words of random access memory.

To accomplish the stand-alone task of current control and data analysis, it is most convenient to use hard disks for these large programs and data files. The two disk drives use two 5.2 megabyte RL01 (1) hard disks. The other hardware installed for data analysis are a CRT terminal with graphics capability and a dedicated plotter. A second CRT is used to display messages from the control program, and a line printer is installed to aid in program development and tabulate data. The CRTs and line printer are interfaced with the computer through a four-port asynchronous serial card, set at 9600 BAUD rate for the terminals and 1200 BAUD rate for the hardcopy plotter.

The control functions described in the current controller section are supplied by two 4-channel D/A converters (5) for the 0 to 10-volt set point, and by two 16-bit parallel interface cards (6) for the logic control lines. The input ports of one of the 16-bit parallel interface cards are used to determine if the controller is in the appropriate mode to receive computer commands. At the time that this reading is made, the controller disconnect timer is reset by an 0.5 ms pulse from the "data ready" line of the interface card. This is a convenient method to

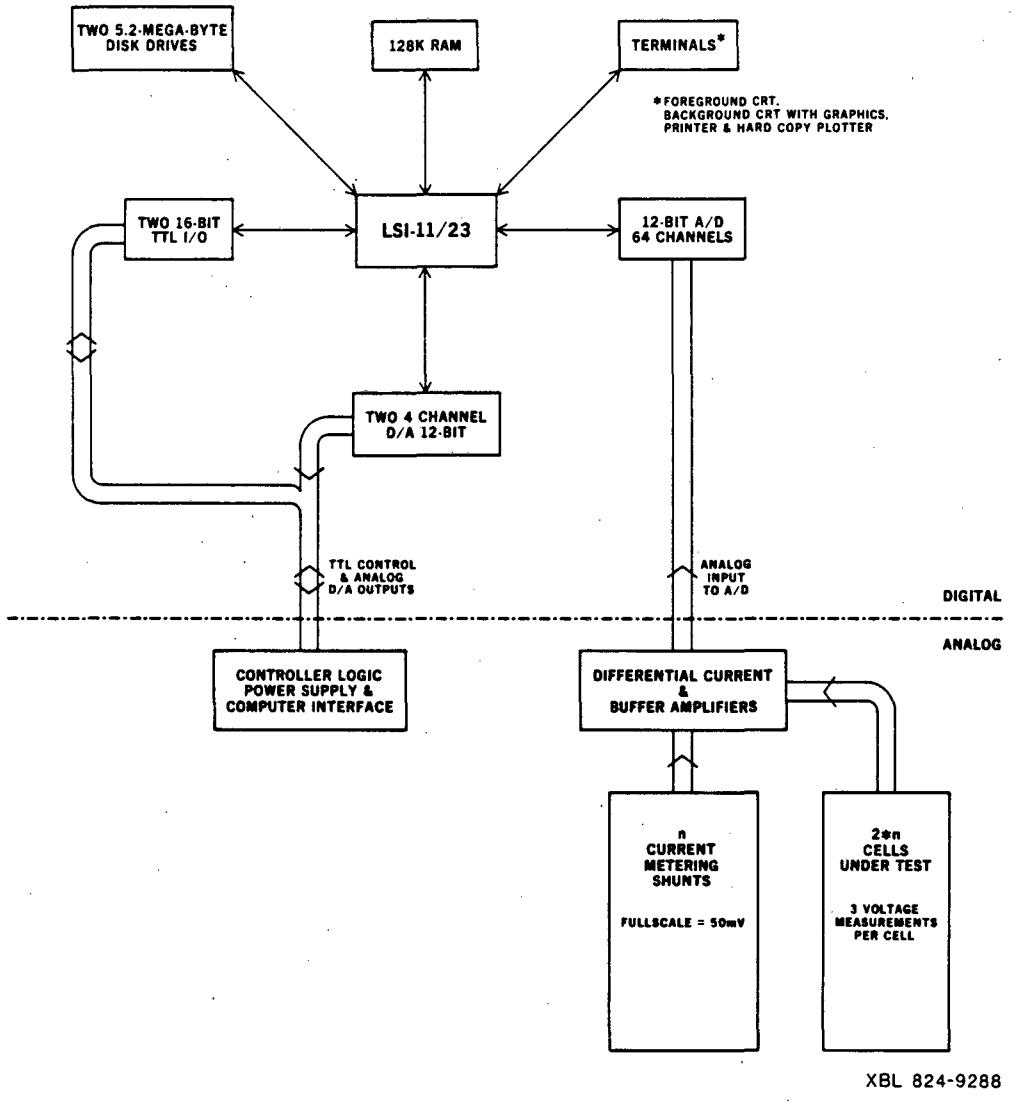


FIGURE 7. Block Diagram for the LSI-11/23 Computer System and Peripherals.

prevent inadvertent damage to the cells should the software program fail to control the current.

To identify the end of charge or improper cell voltages, the software program must be able to continuously monitor the cell voltages. This capability is provided through a 64 channel A/D converter (7) using two 8-channel master cards and two 24-channel slave cards. The current level and polarity are also monitored through the use of the A/D converters, a precision shunt, and the previously-described amplifiers.

REFERENCES

1. Digital Equipment Corporation, Maynard, Massachusetts.

2. NIM - Nuclear Instrumentation Module; see Standard Nuclear Instrument Modules adopted by AEC NIM Committee, TID-20893 (Rev. 4), July 1974.

3. Control and metering shunts are of 0.25% absolute accuracy and are manufactured by Weston, Westinghouse, or Empro.

4. VN1206N1 or VN1204N1, N-channel enhancement mode power MOSFET. Supertex Inc., Sunnyvale, California.

5. DT2766. Data Translation, Natick, Massachusetts.

6. DT2768. Data Translation, Natick, Massachusetts.

7. ST-LSI2 and ST-LSI2-ADX. Datel, Mansfield, Massachusetts.

Figure Titles and Captions

---

1. Multiple Electrochemical Cell Test System-Block Diagram.

   An interconnection block diagram for n independent current controllers. Cabling is detailed for each of n controllers with two cells under test in series.

2. Current Controller-Block Diagram.

   Current controller digital and analog circuitry is shown with the relay positioned for the cells under test disconnected. When connected, relay contacts 1A and 1B will be closed. Charge current will flow from the charge supply through Q1, D1, the current metering shunt, the cells under test, the charge control (Q2), and return to the supply common via the control shunt. Discharge current will flow from the discharge supply through Q3, D2, the cells under test, the current metering shunt, the discharge control (Q4), and return to the supply common via the control shunt.

3. Inside View of Current Controller, Logic Card and Control Current Shunt.

   A side view photograph of the current controller which shows the printed circuit logic card. The control shunt may be seen mounted directly to the bottom of the printed circuit logic card.

4. Inside View of Current Controller, Heat Dissipating Elements.

A photographic view of the heat dissipating elements side of the current controller. Note, the charge VMOS switch and discharge control VMOS are on one heat sink, while the discharge VMOS switch and the charge control VMOS are on the other heat sink.

5. Current Controller, Front and Rear Panels.

Local (manual) front panel control of all functions is possible by means of the switches and push buttons shown. Indicators are provided to clearly show the present status of the current controller, even while under computer control. The fuses and relay are mounted on the rear panel for easy access. A 40 conductor flat ribbon cable passageway is provided at the lower left of the rear panel. By means of this cable the current controller is connected to the logic power supply and computer interface.

6. Current Response and Crosstalk Waveforms.

   The top waveform is a 7 volt external function generator set point control voltage. The middle trace is the corresponding 7 ampere charge current waveform. The lower trace is typical of the current change induced in any other current controller output due to the 7 ampere change shown in the middle trace.

7. LSI-11/23 Computer System and Peripherals.

   Interconnection block diagram detailing computer hardware and its interconnections with the current controller and cells under test.

APPENDIX B

MEMORY USAGE AND THE EXTENDED MEMORY MONITOR

This appendix assumes that the reader has a basic understanding of computer systems and a working knowledge of RT-11. The information in the following discussion can be gathered from the RT-11 documentation (1). It is, however, scattered throughout various volumes. The intent of this Appendix is, therefore, to reflect on difficulties encountered in using the extended memory monitor.

A schematic representation of the physical address space of a 128K - system, in the RT-11 environment, is given in Fig. 1 (*). A background job with low-memory overlay structure occupies the memory (2).

Note that none of the memory above 28K is used, with exception of the I/O page. This is always true in the SJ or FB monitor, unless one makes use of the virtual array feature.

The size of the real-time program precludes the use of a low memory overlay structure: the program cannot fit in the lower 28K, either as a background job, or as a foreground job. Therefore, the use of the XM monitor is required, and to take full advantage of the 128K of memory available, one must run virtual jobs. The memory usage of a virtual

---

(*) All the drawings in this Appendix are modeled on the symbolic representations used in the RT-11 documentation.

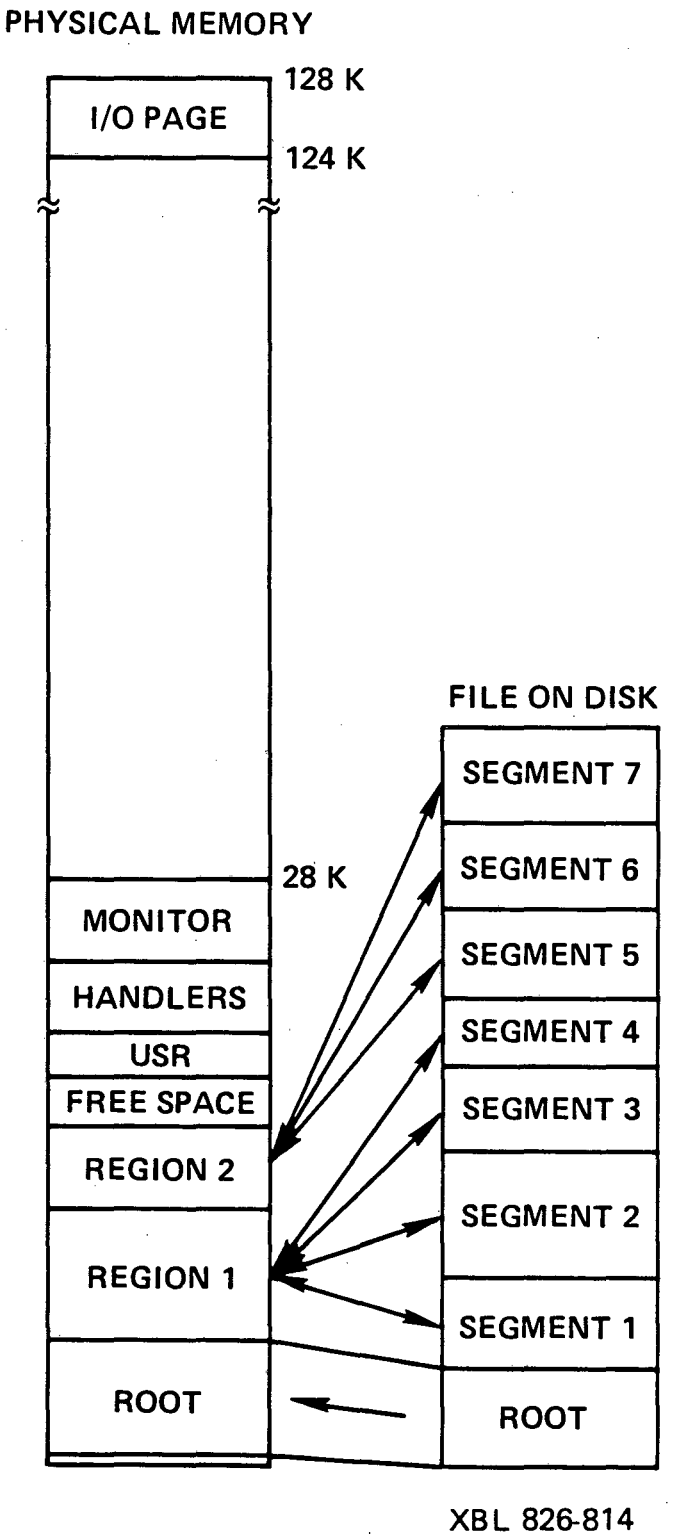


FIGURE 1. Background Program with Low Memory Overlay Structure.

Segments 1, 2, 3, 4 share overlay region 1
Segments 5, 6, 7 share overlay region 2

background job is shown schematically in Fig. 2.

The concepts of virtual jobs, virtual address space (VAS) and memory mapping are explained in the RT-11 documentation (3). With FORTRAN programs, two applications are possible:

1. Declaring arrays virtual in the program. The program is loaded in the lower 28K, and the system uses page 7 of the virtual address space to map the different parts of the arrays (4).

2. Linking the program with a virtual overlay structure. The root of the program is mapped to the lower 28K, and the other VAS-pages serve as overlay regions. In this case, contrary to low-memory overlays, the entire code is loaded in the memory. There is no physical transfer between the memory and a storage device (5).

Note that virtual arrays and virtual overlays can be used at the same time in a virtual job.

There are two important considerations pertaining to the use of memory with a virtual job:

1. The size of the root. This part of the program always resides in the lower 28K with the monitor, the device handlers and the USR. Usually, with the XM monitor, only 18K are available for the user program. This problem is of critical importance when two programs share the memory (e.g., a foreground and a background job, or system jobs): indeed, the root of the two programs together must fit in the available 18K.

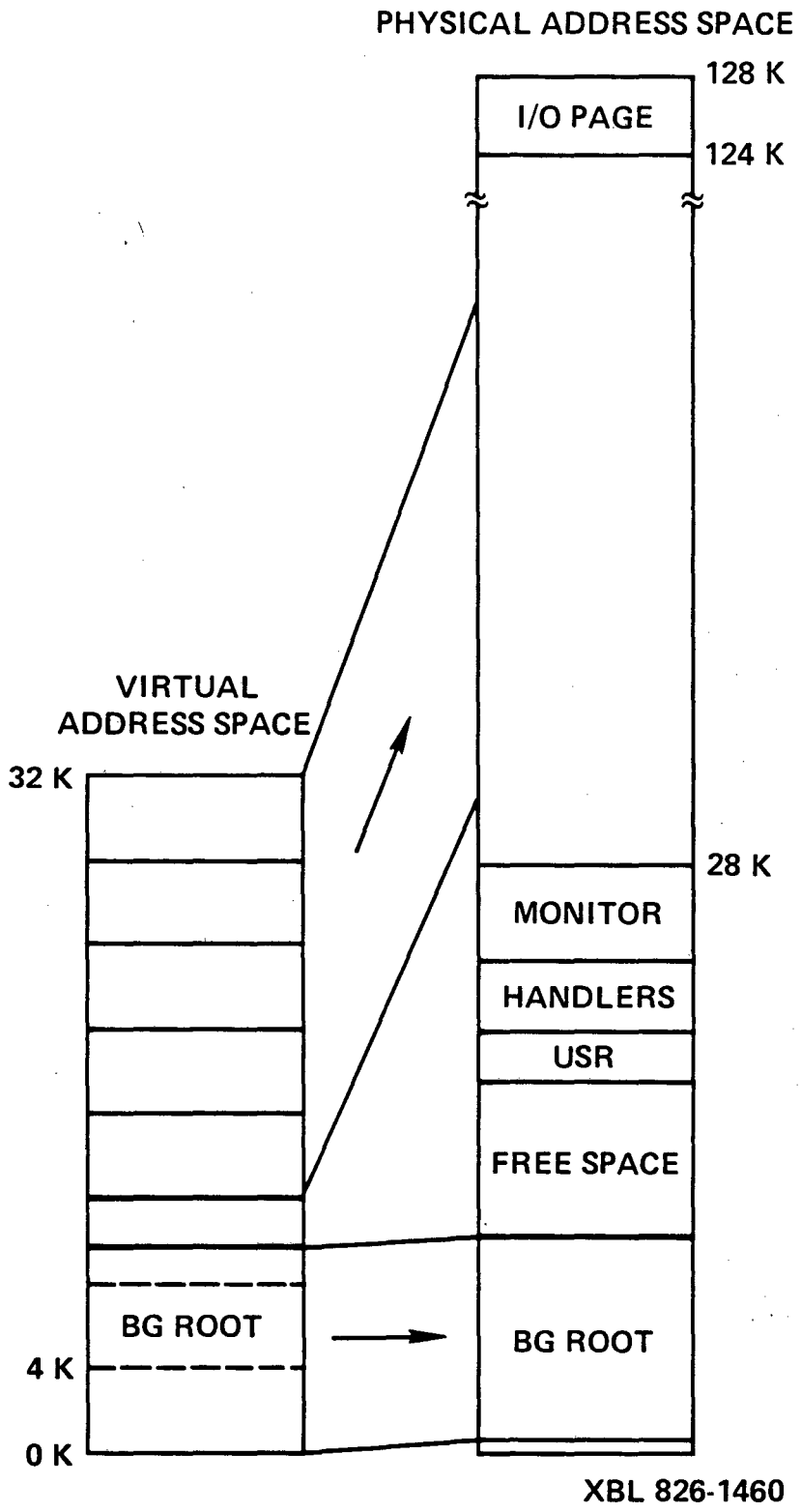


FIGURE 2. Usage of Physical and Virtual Address Space, for a Virtual Background Job.

A few factors which affect the size of a program's root are:

1. The FORTRAN Object Time System (OTS) routines automatically get loaded in the root; they account for a large fraction of its size (when the program is complex) and use many subroutines of the system library.

2. COMMON arrays that are accessed in different overlay regions are loaded in the root and therefore increase its size.

3. Storage space for arrays is allocated to the first routine where they are defined, so it is useful to exclude them from the root. Therefore, a minimal root size can be obtained in the following way

```
PROGRAM START
CALL MAIN
END
```

Program START is defined as the root, and subroutine MAIN (together with all the subroutines it might call) encompasses one or more virtual overlays. MAIN is the actual user program. The root is small, since no local arrays are loaded in it.

4. Channel areas, queue elements, and interrupt service routines must be located in the root, but they may not reside (nor, in the latter case, reference addresses) in the region between 4K and 8K (6).

2. The total size of the virtual address space. The VAS is 32K long, and includes 8 pages, each 4K long. When virtual arrays are used, page 7 automatically becomes unavailable for any other use. In general, the OTS work area takes up another page, so that only 5 or 6 pages are left for the root and all the virtual overlay regions. Since virtual overlay regions always start on 4K boundaries, there will only be a limited number of these regions available. For example, for a job with a 6K root and no virtual arrays, we arrive at the situation shown in Fig. 3.

   In this case, we can have 5 regions smaller than 4K, or 1 region smaller than 4K and 2 regions between 4 and 8K, etc. Since virtual overlay regions are always at least 4K long, one can combine small overlay segments into bigger ones and retain their sizes below 4K. For example

```
SEGM1,SEGM2,SEGM3/V:1
SEGM4/V:1
SEGM5/V:1
```

   Calls between SEGM1, SEGM2 and SEGM3 are legal.

The link commands for the REAL-TIME and WATCHDOG programs are:

```
R LINK
REALTM.SAV,REALTM=START/P:200./W,CRMESS,CRSTOP/V/I//
REALTM,GO,GDNITE,MSBOSS,HDCONC/V:1
GMOPEN,CHANGE,MSWHO,MSRUN/V:2
CHECK,INPUT,TIMEAS/V:2
```

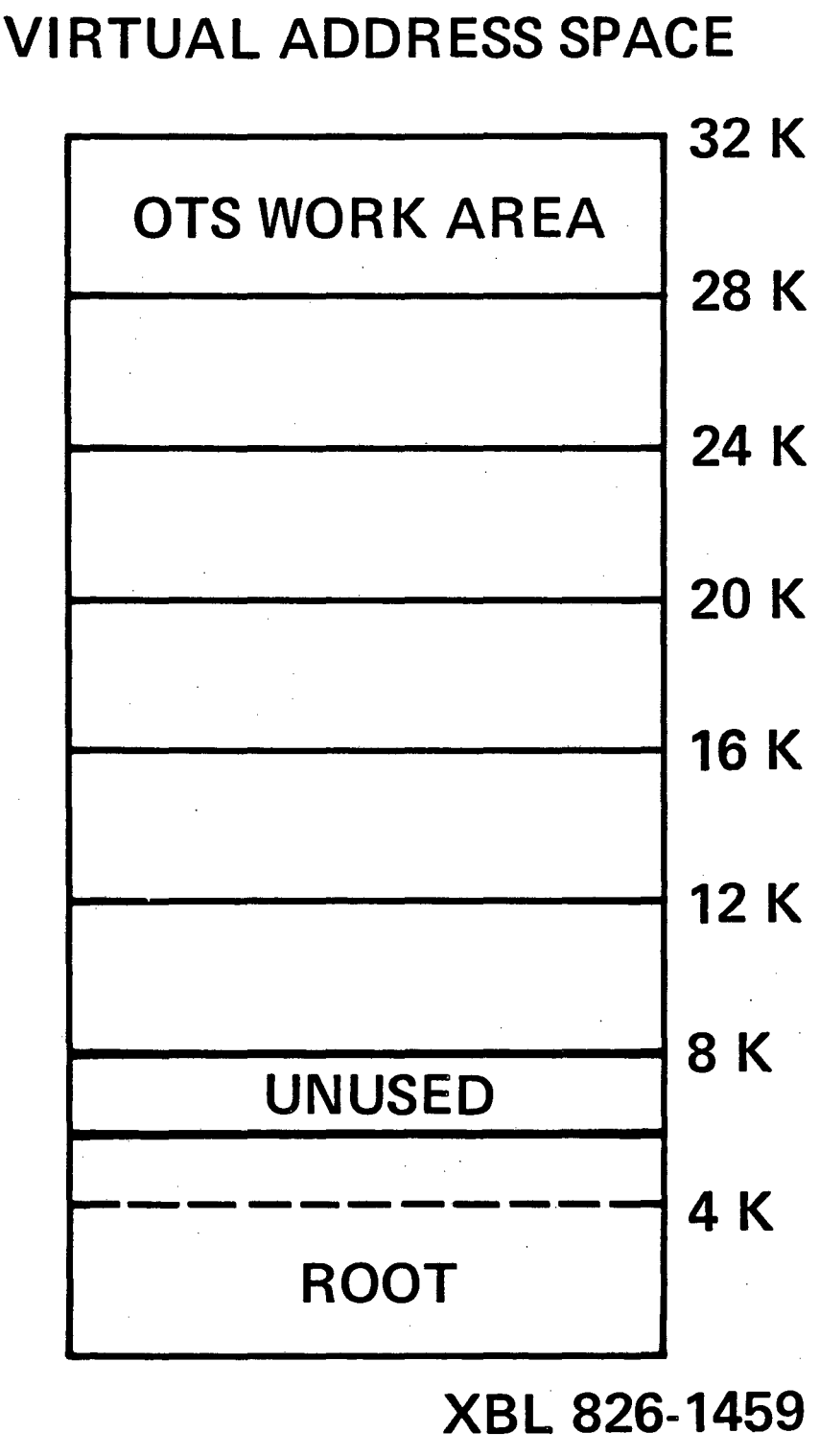


FIGURE 3. Usage of Virtual Address Space for a virtual FORTRAN job. Pages 0 and 1 are used by the root; page 7 is used by the OTS work area; pages 2 to 6 are available for mapping.

```
FINISH,CHOOSE,CC,CVLC/V:2
MSCHNG,MSUPDT,MSWHEN,MSSTOP,MSHOLD,MSNOW/V:2
ANNOUN,IDATE,FILOPC,UPDATE/V:3
CP,CALIBP,EXCON,CALIB,CALIBV/V:3
RESET/V:3
//
$QBLK


R LINK
WATCHD.SAV=WATCHD/V/I//
MATIN/V:1
MODIF/V:1
RUN/V:1
HOLD/V:1
ASK/V:2
//
$QBLK
```

A schematic view of the memory usage of the 2 programs is given in Fig. 4.

A particular restriction which affects the size of the root of a foreground program (such as the REAL-TIME program) is the location of the USR. Under XM, the USR must not be loaded in the 4K to 8K region; however, it does load below the foreground root. Since the USR size is 2K, and (as shown in Fig. 4.) the foreground program's upper limit is around 20K, the maximum size of the foreground root is 10K (even <u>without</u> a background program running!).

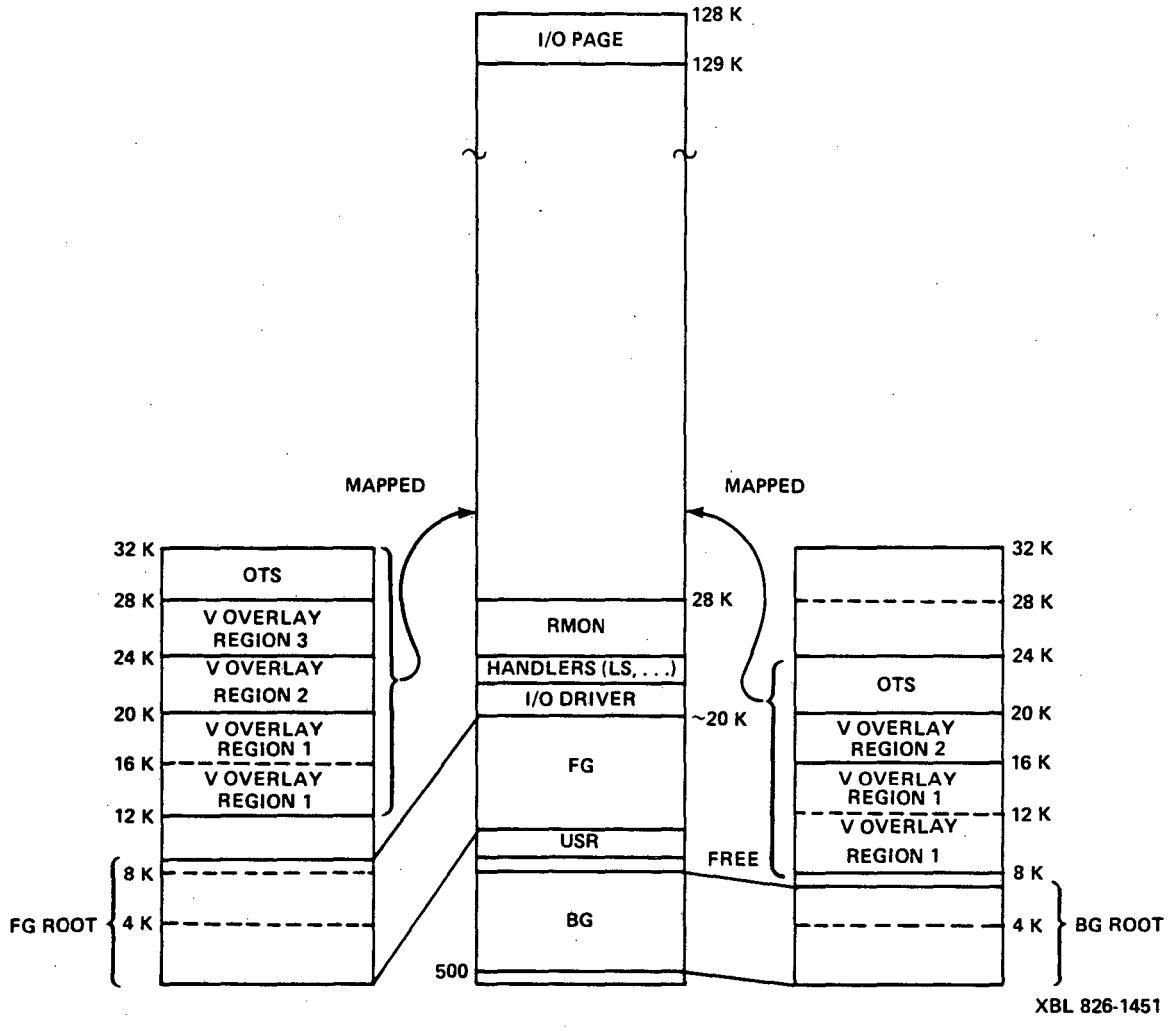


FIGURE 4. Memory Usage for the REAL-TIME (Foreground) and WATCHDOG (Background) programs.

The REAL-TIME program uses 3 virtual overlay regions; the WATCHDOG program uses 2 virtual overlay regions.

The main restriction of virtual jobs is that they cannot access the I/O page (one is not allowed to map any part of the VAS to the monitor or I/O page areas). One way to circumvent this problem is to perform indirect I/O by using a driver. In that case, I/O transfers are conducted under monitor control in system state (as opposed to user state), where access to all parts of the memory is allowed. The I/O driver is described in detail in Appendix C.

REFERENCES

1. RT-11 Documentation Directory, Version 4, Vol. 1., Jan. 1980. Digital Equipment Corporation, Maynard, Massachussetts.

2. RT-11 Documentation, Version 4, Vol. 2a, SUG, pp. 11-18 to 11-26, March 1980. Digital Equipment Corporation, Maynard, Massachussetts.

3. Ibid., Vol. 3B, SSM, pp 4-1 to 4-37.

4. Ibid., Vol. 4, FORTRAN User's Guide, pp 2-3 to 2-8.

5. Ibid., Vol 2A, SUG, pp 11-27 to 11-40, and Vol. 3B, SSM, page 4-35.

6. Ibid., Vol 3B, SSM, page 4-66

Appendix C

THE I/O DRIVER AND THE I/O LIBRARY

C.1 THE I/O DRIVER

C.1.1 CALLING THE DRIVER

The I/O driver is written in Macro-11 assembly language code using the standard instruction set for LSI-11/23 computers (1-3). The name of the driver (or device handler) is HD, and the corresponding system file is HD.SYS for the SJ and FB monitors, and HDX.SYS, for the XM monitor. HD must be loaded in memory, just below the monitor and the LS handler, with the commands INSTALL HD and LOAD HD. HD does not have an interrupt section, which means that it executes an I/O transfer in one continuous sequence (unlike drivers for disk I/O,...). As required for all device handlers, HD is written in position-independent code (PIC).

Access to HD is done through the use of 'special function' calls, which are RT-11 system routines (4). A general procedure for using HD is given in the following program excerpt:

```
      PROGRAM TESTHD
      INTEGER DBLK(4),IBUF(10)
      DATA DBLK/3RHD ,0,0,0/
            .
            .
C     get a channel to communicate with HD
      MCHAN=IGETC()
            .
            .
C     open communication path with HD, using channel MCHAN
      J=LOOKUP(MCHAN,DBLK)
            .
            .
C     call HD
      I=ISPFNW(ICODE,MCHAN,IWCNT,IBUF(1),IBLK)
            .
            .
      END
```

A description of the parameters follows:

ICODE : function code, as an octal number (see next sections)

MCHAN : channel through which the program communicates with HD.

DBLK : device specification for lookup routine.

IWCNT : number of words in IBUF that are used for this function (maximum 10), as described in the next sections.

IBUF(1) : first element of IBUF (=address of IBUF). If a hardware error occurs, it contains a special error code returned by the driver.

IBLK : dummy argument for the ISPFNW call (any integer number).

The system returns a standard error code for ISPFNW

i = 0 : normal - no error

1 : attempt to read or write past end-of-file

2 : hardware error occurred on channel

3 : channel specified is not open

When the driver code detects an error in the input parameters, it notifies the system, which sets I=2. In that case only, the driver itself returns a special error code in IBUF(1).

IBUF(1) = -2 : non-existent function code

-1 : wrong IWCNT for function

0 : normal return

> 0 : specific error code for each function, see next section

The ISPFNW call causes the program execution to halt until the driver completes the transfer. Three other versions of this call can be used: ISPFN, ISPFNC, ISPFNF (but more caution is called for because they do not wait for the driver to finish).

C.1.2 DRIVER FUNCTIONS AND HARDWARE CONNECTIONS

C.1.2.1 TALK Function -

1. Action: CONNECT(ON)/DISCONNECT(OFF) a controller, set charge/discharge

2. Hardware: 1st parallel I/O card, output

3. Octal code: 200

4. Word count: 4

5. Input arguments:

    1. IBUF(2): 1 for ON, 0 for OFF
    2. IBUF(3): 1 for CH, 0 for DS
    3. IBUF(4): current controller (CC) number: 1 to 8

6. Output arguments: none

7. Error codes(IBUF(1))

    1. 1 : IBUF(2) not = 0 or 1
    2. 2 : IBUF(3) not = 0 or 1
    3. 3 : IBUF(4) < 1 or > 8

8. Hardware connections: Bits 0 and 1 of the 16-bit I/O output register go to CC1; bits 2 and 3 to CC2; bits 4 and 5 to CC3; etc.

    for CC1

        bit 0 : ON/OFF bit

            bit set is ON, bit clear is OFF

        bit 1 : CH/DS bit

            bit set is DS, bit clear is CH

C.1.2.2 LISTEN Function -

1. Action: read status of a controller: local or computer mode

2. Hardware: 1st parallel I/O card, input

3. Octal code: 201

4. Word count: 3

5. Input arguments: IBUF(3) : CC number : 1 to 8

6. Output arguments: IBUF(2) : 0 for computer mode, 1 for local mode

7. Error codes(IBUF(1)) : 1 : IBUF(3) < 1 or > 8

8. Hardware connections: Bit 0 of the 16-bit I/O output register comes from CC1; bit 1 from CC2; bit 2 from CC3; etc.

for CC1 : bit 0 : bit set is local,

bit clear is computer mode

C.1.2.3 DAC Function -

1. Action: send a voltage setpoint to a controller

2. Hardware: D/A cards, output

3. Octal code: 202

4. Word count: 3

5. Input arguments:

1. IBUF(2) : CC number : 1 to 8
2. IBUF(3) : integer word : 0 to 4095.

6. Output arguments: none

7. Error codes (IBUF(1)):

    1. 1 : IBUF(2) < 1 or > 8
    2. 2 : IBUF(3) < 0 or > 4095.

8. Hardware connections: 4095 gives the full-scale voltage setpoint, that is 10 V, and it corresponds to the full-scale current rating of the controller (2 A or 10 A). A smaller current is an integer fraction of 4095.

C.1.2.4 ADC Function -

1. Action: read an A/D channel, and return an integer result that is the average of a number of readings.

2. Hardware: A/D cards, input

3. Octal code: 203

4. Word count: 7

5. Input arguments:

    1. IBUF(2) : channel number : 0 to 63.
    2. IBUF(3) : range : 1,2,4,8 (+/- 5 V, +/- 2.5 V, +/- 1.25 V, +/- .625 V)
    3. IBUF(4) : number of average readings for the measurement : 1,2,4,8,16
    4. IBUF(5) : delay count between each of the averaged readings : (= number of times a delay loop is executed) : 0 to highest integer number

6. Output arguments:

    1. IBUF(6) : average result (integer)
    2. IBUF(7) : remainder (= 0 if result is not an average, that is only 1 measurement)

7. Error codes (IBUF(1)):

    1. 1 : IBUF(2) < 0 or > 63.
    2. 2 : IBUF(3) not = 1,2,4,8
    3. 3 : IBUF(4) not = 1,2,4,8,16
    4. 4 : IBUF(5) < 0

8. Hardware connections: see buffer box wiring diagram, Table 1. A master A/D card and a slave expander give a total of 32 differential channels, numbered 0 to 31. There are 2 such pairs of cards. The driver recognizes that A/D channels 32 to 63 are actually channels 0 to 31 of the 2nd pair of cards.

C.1.2.5 EXTERNAL Function -

1. Action: put a controller in normal or external mode

2. Hardware: 2nd parallel I/O card, output

3. Octal code: 204

4. Word count: 4

5. Input arguments:

    1. IBUF(2) : 1 for EXT, 0 for NORMAL
    2. IBUF(3) : dummy argument : 0 or 1
    3. IBUF(4) : CC number : 1 to 8

6. Output arguments: none

7. Error codes (IBUF(1)) :

    1. 1 : IBUF(2) not = 0 or 1
    2. 2 : IBUF(3) not = 0 or 1
    3. 3 : IBUF(4) <1 or >8

8. Hardware connections : Bit 8 of the 16-bit I/O output register goes to CC5; bit 10 to CC6; bit 12 to CC7; and bit 14 to CC8; only these 4 controllers are presently wired up for that function

    for CC5 : bit 8 : bit set is local,

    bit clear is computer mode

C.1.2.6 ADT Function -

1. Action: read an A/D channel for a certain time (see C.1.2.4), and return minimum or maximum value

2. Hardware: A/D cards and programmable clock

3. Octal code: 205

4. Word count: 10

5. Input arguments:

    1. IBUF(2) : channel number : 0 to 63.
    2. IBUF(3) : range : 1,2,4,8 (see C.1.2.4)
    3. IBUF(4) : number of averaged readings for each measurement : 1,2,4,8,16
    4. IBUF(5) : delay count between each of the readings (see C.1.2.4)

5. IBUF(6) : 1 for maximum, 0 for minimum
6. IBUF(7) : total time, in .1 msec units : maximum is 10,000 (= 1 sec)

6. Output arguments:

1. IBUF(8) : integer result (minimum or maximum)
2. IBUF(9) : remainder (see C.1.2.4)
3. IBUF(10) : number of averaged measurements compared for max/min determination (total number of readings = IBUF(10) * IBUF(4))

7. Error codes (IBUF(1)):

1. 1 : IBUF(2) <0 or > 63.
2. 2 : IBUF(3) not = 1,2,4,8
3. 3 : IBUF(4) not = 1,2,4,8,16
4. 4 : IBUF(5) not = 0 or 1
5. 5 : IBUF(6) > 10,000 (time > 1 sec)

8. Hardware connections: see C.1.2.4

C.1.2.7 Test Program For The I/O Driver -

A test program is available to check the I/O functions described above. It is called with the command R TESTHD: the user specifies the octal code number and the appropriate word count, and enters the input arguments. The program returns a line with the input arguments (IBUF(1) to (10)), a line with the ISPFNW return code, and a line with the output arguments (IBUF(1) to (10)).

C.1.3 INSTALLING THE DRIVER

The file that contains the driver code is called HD.MAC. When the driver is modified, it needs to be re-installed: this should be done for the SJ/FB and the XM monitors.

For SJ/FB:

```
MACRO/CROSSREFERENCE/SHOW:MEB DLX:SYCND.DIS+DLX:HD/OBJ/LIST
LINK/EXECUTE:HD.SYS HD/MAP
INSTALL HD
LOAD HD
```

For XM:

```
MACRO/CROSSREFERENCE/SHOW:MEB DLX:(XM+SYCND+HD)/OBJ:HDX/LIST:HDX
LINK/EXECUTE:HDX.SYS HDX/MAP
INSTALL HD
LOAD HD
```

DLX is DL0 or DL1. At link time, a message will appear: 'global HDINT undefined', because there is no interruption section in the driver.

C.1.4 DRIVER CODE

```
.TITLE  HD      V04.01
;HD HANDLER
.IDENT  /V04.01/
.SBTTL  PREAMBLE SECTION
.MCALL  .DRDEF
.DRDEF  HD,377,SPFUN$,0,167770,400
TLK$FN=200              ;TALK code
LSN$FN=201              ;LISTEN code
DAC$FN=202              ;DAC code
ADC$FN=203              ;ADC code
EXT$FN=204              ;EXTERNAL code
ADT$FN=205              ;ADT code
CLK$FN=206              ;CLOCK code, non-existent function
HDPAR1=172342           ;PAR1 register location
```

```
TLKOUT=167772           ;output register of 1st parallel I/O card
LSNIN=167774            ;input register of 1st parallel I/O card
EXTOUT=167762           ;output register of 2nd parallel I/O card
DACOUT=170400           ;output register of DAC card
CLKCSR=170420           ;clock status register
CLKBPR=170422           ;clock buffer/preset register
;MARCO KATZ  december 1981
.SBTTL  HEADER SECTION
.DRBEG  HD
.SBTTL  I/O INITIALIZATION SECTION
        MOV     HDCQE,R4        ;get pointer to current queue element
        MOV     Q$BUFF(R4),R5   ;put buffer address (virtual) in R5
.IF     NE,MMG$T
        MOV     @#HDPAR1,-(SP)  ;save old PAR1 on stack
        MOV     Q$PAR(R4),@#HDPAR1      ;put new value in PAR1 (map to
                                ;buffer)
.ENDC
        CLR     (R5)            ;set error code = 0 (1st word in BUFFER)
        MOVB    Q$FUNC(R4),R3   ;put function number in R3
        BIC     #177400,R3      ;keep bits 0 to 7
F1 :    CMP     R3,#TLK$FN      ;talk function ?
        BNE     F2
        JMP     TLK             ;if yes, branch to it
F2 :    CMP     R3,#LSN$FN      ;listen function ?
        BNE     F3
        JMP     LSN             ;if yes, branch to it
F3 :    CMP     R3,#DAC$FN      ;D/A conversion ?
        BNE     F4
        JMP     DAC             ;if yes, branch to it
F4 :    CMP     R3,#ADC$FN      ;A/D conversion ?
        BNE     F5
        JMP     ADC             ;if yes, branch to it
F5 :    CMP     R3,#EXT$FN      ;external function ?
        BNE     F6
        JMP     EXT             ;if yes, branch to it
F6 :    CMP     R3,#ADT$FN      ;timed A/D conversion ?
        BNE     F7
        JMP     ADT             ;if yes, branch to it
F7 :    CMP     R3,#CLK$FN      ;general clock access ?
        BNE     SPFNER          ;if no, wrong code
;       JMP     CLK             ;if yes, branch to it
                                ;not implemented yet
SPFNER : MOV    #177776,(R5)    ;function code is no good
                                ;error code is -2
        JMP     HDERR
.SBTTL  TALK SECTION
;
;This section of the handler sets a current controller ON/OFF - CH/DS.
;First, it checks for a right number of parameters passed and checks
;if they are consistent with the specifications.
;A controller is set ON/OFF ; AO /DS, by toggling 2 bits in the 16-bit
;I/O output register. Bits 0 & 1 are for CC1 (0 for ON/OFF and 1 for
;CH/DS), bits 2 & 3 are for CC2 (idem) ...
;R4 and R0 indicate what type of action is being taken.
```

```
;The code moves a 1-bit to the correct position and, according to the
;hardware specifications, sets or clears it in the output register:
;               ON, DS  : set (R0=0)
;               OFF, CH : clear (R0=1)
;For OFF, the code only takes care of the ON/OFF bit (R4=0), for
;ON, it first handles the CH/DS bit (R4=1), then the ON/OFF bit (R4=0).
;R5 points to the 1st element of the parameter buffer.
;This section of the handler also sets a controller NORMAL-EXTERNAL,
;when it is called with function "external". Setting NORMAL is like
;setting OFF, setting EXTERNAL is like setting ON.
;
TLK    : MOV     #TLKOUT,R3      ;put out address of 1st I/O card in R3
         BR      TTLK            ;go
EXT    : MOV     #EXTOUT,R3      ;put out address of 2nd I/O card in R3
TTLK   : CMP     Q$WCNT(R4),#4   ;4 words in buffer ?
         BEQ     TTLK1           ;if yes, branch to next test
         MOV     #177777,(R5)    ;error code is -1
         JMP     HDERR
TTLK1  : BIT     #177776,2(R5)   ;is ON/OFF = 0 or = 1 ?
         BEQ     TTLK2           ;if yes, branch to next test
         MOV     #1,(R5)         ;error code is 1
         JMP     HDERR
TTLK2  : BIT     #177776,4(R5)   ;is CH/DS = 0 or = 1 ?
         BEQ     TTLK3           ;if yes, branch to next test
         MOV     #2,(R5)         ;error code is 2
         JMP     HDERR
TTLK3  : CMP     6(R5),#1        ;is CC < 1 ?
         BLT     TTLK3E          ;if yes, branch to error
         CMP     6(R5),#10       ;is CC < or = 8. ?
         BLE     TALK            ;if yes, branch to action
TTLK3E: MOV     #3,(R5)         ;error code is 3
         JMP     HDERR
TALK   : MOV     2(R5),R4        ;move ON/OFF to R4
         MOV     #1,R0           ;move 1 to R0
         TST     R4              ;is R4 = 0 ? (ON/OFF setting)
         BEQ     TLKRPT          ;if yes, branch
         MOV     4(R5),R0        ;if no, move CH/DS to R0
TLKRPT: MOV     6(R5),R1        ;move CC # to R1
         ASL     R1              ;multiply by 2 : we now have a counter
         MOV     #1,R2           ;move 1 to R2 (bit in far-right postn.)
         JSR     PC,FORM         ;jump
         TST     R0              ;is R0 = 0 ?
         BNE     OFF             ;if not, we need to clear the bit
ON     : BIS     R2,(R3)         ;if yes, we need to set the bit
         BR      CHECK           ;branch
OFF    : BIC     R2,(R3)         ;clear bit
CHECK  : TST     R4              ;is R4 = 0 ?
         BGT     MODE            ;if not, continue
         JMP     HDDONE          ;branch to I/O completion section
MODE   : DEC     R4              ;make R4 = 0 (ON/OFF setting)
         CLR     R0              ;clear R0 (prepare for ON setting)
         BR      TLKRPT          ;go and go it
FORM   : TST     R4              ;is R4 = 0 ? (ON/OFF setting)
         BEQ     NEXT            ;if yes, branch
```

```
        DEC     R1              ;decrement counter
NEXT  : DEC     R1              ;decrement counter
        BEQ     TLKDON          ;if counter 0, bit in right position
        ASL     R2              ;if not, move bit to the left
        BR      NEXT            ;repeat
TLKDON: RTS     PC              ;return to main routine
.SBTTL  LISTEN SECTION
;
;This section of the handler checks the status of a current
;controller (computer/local).
;First, it checks for a right number of parameters passed and
;checks if they are consistent with the specifications.
;The code looks at one bit of the 16-BIT I/O register : bit 0
;is for CC1 (clear is computer, set is local), bit 1 for CC2,...
;R5 points to the 1st element of the parameter buffer.
;
LSN   : CMP     Q$WCNT(R4),#3   ;3 words in buffer ?
        BEQ     TLSN1           ;if yes, branch to next test
        MOV     #177777,(R5)    ;error code is -1
        JMP     HDERR
TLSN1 : CMP     4(R5),#1        ;is CC < 1 ?
        BLT     TLSN1E          ;if yes, branch to error
        CMP     4(R5),#10       ;is CC < or = 8. ?
        BLE     LISTEN          ;if yes, branch to action
TLSN1E: MOV     #1,(R5)         ;error code is 1
        JMP     HDERR
LISTEN: MOV     @#LSNIN,R0      ;move input register to R0
        MOV     4(R5),R1        ;move CC # to R1
LSNRPT: DEC     R1              ;decrement R1
        BEQ     LSNDON          ;branch if bit for this
                                ;controller (in R0)
                                ;is in far-right position
        ASR     R0              ;if not yet, rotate R0 to the right
        BR      LSNRPT          ;do it again
LSNDON: BIC     #177776,R0      ;keep only LSB of R0
        MOV     R0,2(R5)        ;move R0 to output argument
        JMP     HDDONE          ;branch to I/O completion section

.SBTTL  DAC SECTION
;
;This section of the handler writes to 1 of the 8 channels
;of the D/A converter.
;First, it checks for a right number of parameters passed
;and checks if they are consistent with the specifications.
;R5 points to the 1st element of the parameter buffer.
;
DAC   : CMP     Q$WCNT(R4),#3   ;3 words in buffer ?
        BEQ     TDAC1           ;if yes, branch to next test
        MOV     #177777,(R5)    ;error code is -1
        JMP     HDERR
TDAC1 : CMP     2(R5),#1        ;ch # < 1 ?
        BLT     TDAC1E          ;if yes, branch to error
        CMP     2(R5),#10       ;ch # < or = 8 ?
        BLE     TDAC2           ;if yes, branch to next test
```

```
TDAC1E: MOV     #1,(R5)         ;error code is 1
        JMP     HDERR
TDAC2 : TST     4(R5)           ;integer word < 0 ?
        BLT     TDAC2E          ;if yes, branch to error
        CMP     4(R5),#4095.    ;integer word < or = 4095. ?
        BLE     DACGO           ;if yes, branch to action
TDAC2E: MOV     #2,(R5)         ;error code is 2
        JMP     HDERR
DACGO : MOV     2(R5),R0        ;put ch # in R0
        DEC     R0              ;reduce ch 1-basis to ch 0-basis
        ADD     R0,R0           ;ADDRESS=DACOUT+2*CH#
        ADD     #DACOUT,R0      ;R0 = address of ch
        MOV     4(R5),R1        ;R1 = integer word
;DATA TRANSLATION DAC REQUIRES COMPLEMENT DIGITAL WORD
        COM     R1              ;complement integer word
        MOV     R1,@R0          ;write integer word to ch
        JMP     HDDONE          ;branch to I/O completion section
.SBTTL  ADC SECTION
;
;This section of the handler reads 1 of the 64 A/D channels
;(0 to 63). It averages 1,2,4,8 or 16 readings (with a delay
;between the readings). First, it checks for a right number
;of parameters passed and checks if they are consistent with
;the specifications.  There are 2 pairs of A/D cards: each
;pair has a total of 32 channels (0 to 31.).  The channel
;number is treated as follows: bits 0 to 4 are decoded, which
;gives a channel number from 0 TO 31, then bits 5 & 6 are
;decoded to see if the channel in question is on the 1st or
;2nd pair of A/D cards, so that the right A/D registers are used.
;The gain number is added to a base address, resulting in a
;pointer to a byte that contains the correct octal gain word
;(0 for gain 1, 100 for 2, 200 for 4, 300 for 8).
;Then channel and gain are or-ed in the same register and output
;to the appropriate command/status register.
;After each reading, the code executes a short loop, for a
;number of times equal to the delay count.  When all readings
;are taken, the average is computed: first, the remainder of
;the division is found, then the division is performed
;2, 4, 8, 16 times.
;R5 points to the first element of the parameter buffer.
;part of this section also serves for function ADT.
;
ADC   : CMP     Q$WCNT(R4),#7   ;7 words in buffer ?
        BEQ     TADC1           ;if yes, branch to next test
        MOV     #177777,(R5)    ;error code IS -1
        JMP     HDERR
TADC1 : BIT     #177700,2(R5)   ;is ch # > or = 0 and < or = 63. ?
        BEQ     TADC2           ;if yes, branch to next test
        MOV     #1,(R5)         ;error code is 1
        JMP     HDERR
TADC2 : CMP     4(R5),#10       ;is range > 8 ?
        BGT     TADC2E          ;if yes, branch to error
        MOV     4(R5),R0        ;put range in R0
        JSR     PC,PWR2         ;test for range 1 or power of 2
```

```
        TST     R0              ;is there an error ?
        BEQ     TADC3           ;if no, branch to next test
TADC2E: MOV     #2,(R5)         ;error code is 2
        JMP     HDERR
TADC3 : CMP     6(R5),#20       ;is average count > 16. ?
        BGT     TADC3E          ;if yes, branch to error
        MOV     6(R5),R0        ;put average count in R0
        JSR     PC,PWR2         ;test for average count 2 or power of 2
        TST     R0              ;is there an error ?
        BEQ     TADC4           ;if not, branch to next test
TADC3E: MOV     #3,(R5)         ;error code is 3
        JMP     HDERR
TADC4 : TST     10(R5)          ;is delay count > or = 0 ?
        BGE     ADCGO           ;if yes, branch to action
        MOV     #4,(R5)         ;error code is 4
        JMP     HDERR
ADCGO : MOV     R3,-(SP)        ;put function code on stack
        MOV     2(R5),R4        ;move ch # to R4
        MOV     R4,R2           ;copy it to R2
        BIC     #177740,R4      ;keep only bits 0 to 4 in R4 (0. to 31.)
        ROR     R2              ;rotate R2 to the right until bit 5
        ROR     R2              ;falls on bit 1 and bit 6 on bit 2.
        ROR     R2              ;These 2 bits give an offset-value (0,2)
        ROR     R2              ;that is used to get the address of the
                                ;right A/D registers (card 1,2)
        BIC     #177771,R2      ;keep bits 1 & 2 in R2
        MOV     PC,R3           ;PIC code
        ADD     #CDSTAT-.,R3    ;put address of the command/status
        ADD     R2,R3           ;register (for this channel) in R2.
        MOV     (R3),R2         ;the gain/ch & A/D data register (for
                                ;this channel) is 2+(R2)
        MOV     4(R5),R0        ;move gain to R0
        ADD     PC,R0           ;PIC code
        ADD     #GAINAD-.,R0    ;add base address of gain words to R0
        MOVB    (R0),R1         ;transfer byte, with right bit settings
                                ;for the required gain, to R1
        BIC     #177400,R1      ;clear high byte of R1
                                ;(necessary because of sign extension
                                ;in previous operation)
        BIS     R1,R4           ;set these bits in R4
        CMP     (SP),#ADT$FN    ;ADT function ?
        BEQ     ADTGO           ;if yes, branch to ADT section
ADTRPT: MOV     6(R5),R0        ;move # of readings to R0
        CLR     R3              ;initialize result-register
ADCRPT: CLR     (R2)            ;load program controller mode in
                                ;command register (see DATEL hardware)
        MOV     10(R5),R1       ;move # of delay counts to R1
        MOV     R4,2(R2)        ;load MUX address with channel and range
LOOP  : TST     (R2)            ;end of conversion ?
        BPL     LOOP            ;if no, loop
        ADD     2(R2),R3        ;update result-register
        DEC     R0              ;all measurements taken ?
        BEQ     ADTADC          ;if yes, jump
WAIT  : DEC     R1              ;decrement time-lapse counter
```

```
        BGT     WAIT            ;if > 0, wait
        BR      ADCRPT          ;jump for more readings
ADTADC: CMP     (SP),#ADT$FN    ;ADT function ?
        BEQ     ADTCHK          ;if yes, branch to ADT section
RES   : MOV     6(R5),R0        ;move # of meas. to R0
        MOV     R3,R4           ;copy R3 to R4
        NEG     R0              ;create a masking word that will give
                                ;the remainder OF R3/R0
        BIC     R0,R4           ;get remainder
        NEG     R0              ;restore R0
AGAIN : ASR     R0              ;shift R0 to the right
        BEQ     ADCDON          ;if 0, jump
        ASR     R3              ;shift R3 to the right (R3/2)
        BR      AGAIN           ;do it again
ADCDON: CMP     (SP),#ADT$FN    ;ADT function ?
        BEQ     ADTDON          ;if yes, branch to ADT section
        MOV     R3,12(R5)       ;move quotient to output argument
        MOV     R4,14(R5)       ;move remainder    "        "
        TST     (SP)+           ;restore stack
        JMP     HDDONE          ;branch to I/O completion section
GAINAD: .WORD   0               ;high byte is gain 1
        .WORD   000100          ;low byte is gain 2
        .WORD   000200          ;low byte is gain 4
        .WORD   0
        .WORD   000300          ;low byte is gain 8
CDSTAT: .WORD   170440          ;command/status register, card 1
                                ;gain/channel & A/D data register for
                                ;card 1 is 170442
        .WORD   170450          ;idem, card 2
                                ;idem, card 2 (170452)
.SBTTL  ADT SECTION
;
;This section of the handler reads an A/D channel (see ADC
;section for specifications) continuously for 0 to 10,000
;times .1 msec, and returns the maximum or minimum of all
;recorded measurements.  First, it ;checks for a right number
;of parameters passed and checks if they are consistent with
;the specifications.  In ADC, measurements are taken by groups
;of 1,2,4,8 or 16.  This is also the case here, but no average
;value is computed until the end.  R5 points to the first
;element of the parameter buffer.  This function shares a lot
;of code with function ADC.
;
;This function shares a lot of code with function ADC.
;
ADT   : CMP     Q$WCNT(R4),#12  ;10. words in buffer ?
        BEQ     TADT1           ;if yes, branch to next test
        MOV     #177777,(R5)    ;error code is -1
        JMP     HDERR
TADT1 : BIT     #177776,12(R5)  ;is min/max = 0 or 1 ?
        BEQ     TADT2           ;if yes, branch to next test
        MOV     #5,(R5)         ;error code is 5
        JMP     HDERR
TADT2 : CMP     14(R5),#10000.  ;is time > 1 sec. ?
```

```
        BGT     TADT2E          ;if yes, branch to error
        TST     14(R5)          ;is time > 0 ?
        BLE     TADT2E          ;if no, branch to error
        JMP     TADC1           ;branch to tests in ADC section
TADT2E: MOV     #6,(R5)         ;error code is 6
        JMP     HDERR
ADTGO : TST     12(R5)          ;are we dealing with a lower limit ?
        BEQ     ADTMIN          ;if yes, branch
        MOV     #100000,ADTRES  ;put small neg. # in result storage loc.
        BR      ADTNXT          ;branch to next action
ADTMIN: MOV     #077777,ADTRES  ;put large pos. # in result storage loc.
ADTNXT: MOV     14(R5),R0       ;put # of clock counts in R0
        NEG     R0              ;negate R0
        MOV     R0,@#CLKBPR     ;load buffer/preset register
        CLR     ADTNUM          ;clear counter for # of measurements
        MOV     #31,@#CLKCSR    ;start clock for single interval
        BR      ADTRPT          ;branch to ADC section for A/D conv.
ADTCHK: TST     12(R5)          ;are we dealing with a lower limit ?
        BEQ     ADTLOW          ;if yes, branch
        CMP     R3,ADTRES       ;Is result > maximum ?
        BLE     ADTTIM          ;if no, branch for time check
        BR      ADTOK           ;if yes, branch for result update
ADTLOW: CMP     R3,ADTRES       ;IS result < minimum ?
        BGT     ADTTIM          ;if no, branch for time check
ADTOK : MOV     R3,ADTRES       ;update result
ADTTIM: INC     ADTNUM          ;increment counter for # of meas.
        TSTB    @#CLKCSR        ;time elapsed ?
        BPL     ADTRPT          ;if no, branch back to ADC section
        MOV     ADTRES,R3       ;put result in R3
        BR      RES             ;branch to ADC section to compute final
                                ;result (quotient+remainder)
ADTDON: MOV     R3,16(R5)       ;move quotient to output argument
        MOV     R4,20(R5)       ;move remainder     "       "
        MOV     ADTNUM,22(R5)   ;move # OF meas.    "       "
        TST     (SP)+           ;restore stack
        JMP     HDDONE          ;branch to I/O completion section
ADTRES: .BLKW   1               ;storage location for result
ADTNUM: .BLKW   1               ;# OF measurements
;
;Subroutine PWR2 checks if the contents of R0 is 1 or a
;power of 2, in which case it returns R0 = 0.  Otherwise,
;it returns R0 not = 0.
;
PWR2  : CLC                     ;clear carry
        ROR     R0              ;rotate R0 to the right
        BCS     CARSET          ;if carry bit set, branch
        BNE     PWR2            ;if not set, and R0 not = 0,
                                ;not done yet
                                ;if not set, and R0 = 0, then
                                ;original content of R0 was 0
        INC     R0              ;set error flag
CARSET: RTS     PC              ;if what remains of R0 is not = 0,
                                ;then original content was not a power
                                ;of 2. return
```

```
.SBTTL  I/O COMPLETION SECTION
HDERR : MOV     HDCQE,R4        ;get pointer to current queue element
        BIS     #HDERR$,@Q$CSW(R4)      ;set error bit in CSW
HDDONE: TST     R5              ;do nothing
.IF     NE,MMG$T
        MOV     (SP)+,@#HDPAR1  ;restore PAR1
.ENDC
        .DRFIN  HD              ;jump to monitor
.SBTTL  HANDLER TERMINATION SECTION
        .DREND  HD
.END
```

C.2 THE I/O LIBRARY

The I/O library is called HDLIB, and it consists of 7 routines: SET, GET, PLACE, IMPOSE, RLOGG, EXT, RMLOGG. A user program which uses the library must be linked with HDLIB, and must have the following commands in its main routine (see C.1.1). The files containing the separate routines are named SET.HD, GET.HD, etc..., and the library file is HDLIB.OBJ.

```
      PROGRAM MAIN
      COMMON/HNDLER/MCHAN
      INTEGER DBLK (4)
      DATA DBLK (4)
      DATA DBLK/3RHD ,0,0,0/
               .
               .
               .
C      get a channel and open communication path with Hd
       MCHAN=IGETC()
       J=LOOKUP (MCHAN,DBLK)
               .
               .
C      call to any library function
               .
               .
       END
```

C.2.1 SET

```
CALL SET(IONOFF,IMODE,ICC)
```

1. Action: connect/disconnect current controller, set charge/discharge (see C.1.2.1 - TALK function). Type or print a message.

2. Input parameters:

   1. IONOFF : 1 for on, 0 for off
   2. IMODE : 1 for ch, 0 for ds
   3. ICC : controller number, 1 to 8

3. Output parameters: none

4. Error codes: none

C.2.2 GET

CALL GET(ICOM,ICC)

1. Action: read the status of a controller, local or computer mode (see C.1.2.2 - LISTEN function)

2. Input parameters: ICC : current controller number, 1 to 8

3. Output parameters: ICOM : 0 for computer mode, 1 for local mode.

4. Error codes: none.

C.2.3 PLACE

CALL PLACE(ICC,CURR,IFULL,IERR)

1. Action: set a current controller current (see C.1.2.3 - DAC function), type or print a message.

2. Input parameters:

    1. ICC : current controller number, 1 to 8
    2. CURR : value of current, in A (real number)
    3. IFULL : full-scale current for controller (integer number)

3. Output parameters: none

4. Error codes (IERR):

    1. 0 : no error, current set
    2. 1 : non-existent controller number, or non-matching (controller and full-scale current)
    3. 2 : current requested lower than 0.
    4. 3 : current requested higher than full-scale

C.2.4 IMPOSE

CALL IMPOSE(ICC,CURR,IFULL,IERR)

This subroutine is identical to PLACE, with the difference that no message is typed or printed.

C.2.5 RLOGG

RESULT=RLOGG(ICHAN,IRAN,IAV,IDEL)

1. Action: read an A/D channel, and return a result that is the average of a number of readings (see C.1.2.4 - ADC function)

2. Input parameters:

    1. ICHAN : A/D channel number, see Fig.
    2. IRAN : A/D range : 1,2,4,8 (+/- 5 V, +/- 2.5 V, +/- 1.25 V, +/- .625 V)
    3. IAV : number of averaged readings for a measurement : 1,2,4,8,16. If entered wrong, it is set to 1.
    4. IDEL : delay counts between each of the average readings (=number of times a delay loop is executed) : 0 to highest integer. If entered wrong, it is set to 0.

3. Output parameters: none -- function result RLOGG. 0 is returned if a wrong channel or range number are specified.

4. Error codes: none

5. Note: All results returned are voltages, including measurements on current channels. An additional conversion is needed in these cases, taking into account that current measurements are made on 50 mV shunts, and that these signals are amplified by a factor 100.

Therefore: CURR = (RESULT (V) / 5 (V)) * IFULL

with IFULL the full-scale current of the controller.

C.2.6 EXT

CALL EXT(IEXT,ICC)

1. Action: set a controller in normal or external mode (see C.1.2.5 - EXTERNAL function). Type or print message.

2. Input parameters:

    1. IEXT : 1 for external, 0 for normal
    2. ICC : current controller number, 1 to 8

3. Output parameters: none

4. Error codes: none

C.2.7 RMLOGG

RESULT=RMLOGG(ICHAN,IRAN,IAV,IDEL,MAXMIN,ITIME)

1. Action: read an A/D channel for a certain time, and return minimum or maximum value (see C.1.2.6. - ADT function ; see C.2.5 - RLOGG)

2. Input parameters:

    1. ICHAN, IRAN, IAV, IDEL : see C.2.5 - RLOGG
    2. MAXMIN : 1 for maximum, 0 for minimum
    3. ITIME : total time in .1 msec units : maximum is 10,000 (= 1 sec)

3. Output arguments: none -- function result RMLOGG. 0 is returned if ICHAN, IRAN, MAXMIN or ITIME are incorrectly specified.

4. Error codes: none

5. Note: see C.2.5

C.2.8 I/O LIBRARY CODE

Notes:

1) Some of the routines type (CRT) or print (line-printer) messages. They work with a logical unit NN, equal to 30. Before running a program that uses the I/O library, the user must assign that logical unit either to the CRT (TT) or the line-printer (LS).

2) The routines are not commented : the code is very simple and written on the specifications given in this Appendix.

```
        SUBROUTINE SET(IONOFF,IMODE,ICC)
        COMMON/HNDLER/MCHAN
        DIMENSION IBUF(4)
        IBUF(2)=IONOFF
        IBUF(3)=IMODE
        IBUF(4)=ICC
        NN=30
        WRITE(NN,1000) ICC
1000    FORMAT(1X,'SET CURRENT CONTROLLER ',I2,' IN ')
        GOTO(5,10)IONOFF+1
5       WRITE(NN,2000)
2000    FORMAT('+',T31,'DISCONNECT MODE')
        GOTO 25
10      GOTO(15,20)IMODE+1
15      WRITE(NN,3000)
3000    FORMAT('+',T31,'DISCHARGE MODE')
        GOTO 25
20      WRITE(NN,4000)
4000    FORMAT('+',T31,'CHARGE MODE')
25      II=ISPFNW("200,MCHAN,4,IBUF(1),10)
        RETURN
        END



        SUBROUTINE GET(ICOM,ICC)
        COMMON/HNDLER/MCHAN
        DIMENSION IBUF(3)
        NN=30
        IBUF(3)=ICC
        II=ISPFNW("201,MCHAN,3,IBUF(1),10)
        ICOM=IBUF(2)
D       GOTO 1
        GOTO 15
1       CONTINUE
D       WRITE(NN,1000)ICC
1000    FORMAT(1X,'CURRENT CONTROLLER ',I2,' IS IN ')
D       GOTO(5,10)ICOM+1
5       WRITE(NN,1100)
1100    FORMAT('+',T30,'COMPUTER MODE')
D       RETURN
10      WRITE(NN,1200)
1200    FORMAT('+',T30,'LOCAL MODE')
15      RETURN
        END



        SUBROUTINE PLACE(ICC,CURR,IFULL,IERR)
        COMMON/HNDLER/MCHAN
        DIMENSION IBUF(3)
        NN=30
        IBUF(2)=ICC
        IERR=0
```

```
        IF(ICC.GT.4.AND.IFULL.NE.10) GOTO 15
        IF(ICC.LT.5.AND.IFULL.NE.2) GOTO 15
        VOLT=(CURR/IFULL)*4095.
        IVOLT=VOLT
        IF((VOLT-IVOLT).GT..5) IVOLT=IVOLT+1
        IBUF(3)=IVOLT
        IF(IVOLT.GT.4095)GOTO 5
        IF(IVOLT.LT.0)GOTO 10
        II=ISPFNW("202,MCHAN,3,IBUF(1),10)
        WRITE(NN,500)ICC,CURR
500     FORMAT(1X,'CURRENT CONTROLLER MODULE # ',I2,/,
        11X,'CURRENT IS SET TO ',F10.5,' A')
        RETURN
5       WRITE(NN,1000)
1000    FORMAT(1X,'CURRENT HIGHER THAN FULL-SCALE')
        REWIND NN
        IERR=3
        RETURN
10      WRITE(NN,2000)
2000    FORMAT(1X,'CURRENT LESS THAN ZERO')
        REWIND NN
        IERR=2
        RETURN
15      WRITE(NN,3000) ICC,IFULL
3000    FORMAT(1X,'NON-MATCHING CONTROLLER # (',I1,') AND RANGE (',I2,
        1' A)')
        REWIND NN
        IERR=1
        RETURN
        END



        SUBROUTINE IMPOSE(ICC,CURR,IFULL,IERR)
        COMMON/HNDLER/MCHAN
        DIMENSION IBUF(3)
        IBUF(2)=ICC
        IERR=0
        IF(ICC.GT.4.AND.IFULL.NE.10) IERR=1
        IF(ICC.LT.5.AND.IFULL.NE.2) IERR=1
        IF(IERR.NE.0) RETURN
        VOLT=(CURR/IFULL)*4095.
        IVOLT=VOLT
        IF((VOLT-IVOLT).GT..5) IVOLT=IVOLT+1
        IBUF(3)=IVOLT
        IF(IVOLT.GT.4095) IERR=3
        IF(IVOLT.LT.0) IERR=2
        IF(IERR.NE.0) RETURN
        II=ISPFNW("202,MCHAN,3,IBUF(1),10)
        RETURN
        END
```

```
      FUNCTION RLOGG(ICHAN,IRAN,IAV,IDEL)
      COMMON/HNDLER/MCHAN
      DIMENSION IBUF(7)
      IF(IAV.NE.1.AND.IAV.NE.2.AND.IAV.NE.4.AND.IAV.NE.8.
     1AND.IAV.NE.16) IAV=1
      IF(IDEL.LT.0.OR.IDEL.GT.32000) IDEL=0
      IBUF(2)=ICHAN
      IBUF(3)=IRAN
      IBUF(4)=IAV
      IBUF(5)=IDEL
      IBUF(6)=0
      IBUF(7)=0
      II=ISPFNW("203,MCHAN,7,IBUF(1),10)
      IRES=IBUF(6)
      IREM=IBUF(7)
      RLOGG=(FLOAT(IRES)+FLOAT(IREM)/FLOAT(IAV)+2048.)*(10./IRAN)
     1/4096.-5./IRAN
      RETURN
      END



      SUBROUTINE EXT(IEXT,ICC)
      COMMON/HNDLER/MCHAN
      DIMENSION IBUF(4)
      IBUF(2)=IEXT
      IBUF(3)=0
      IBUF(4)=ICC
      NN=30
      WRITE(NN,1000) ICC
1000  FORMAT(1X,'SET CURRENT CONTROLLER ',I2,' IN ')
      GOTO(5,10) IEXT+1
5     WRITE(NN,2000)
2000  FORMAT('+',T31,'NORMAL MODE')
      GOTO 20
10    WRITE(NN,3000)
3000  FORMAT('+',T31,'EXTERNAL MODE')
20    II=ISPFNW("204,MCHAN,4,IBUF(1),10)
      RETURN
      END



      FUNCTION RMLOGG(ICHAN,IRAN,IAV,IDEL,MAXMIN,ITIME)
      COMMON/HNDLER/MCHAN
      DIMENSION IBUF(10)
      IF(IAV.NE.1.AND.IAV.NE.2.AND.IAV.NE.4.AND.IAV.NE.8.
     1AND.IAV.NE.16) IAV=1
      IF(IDEL.LT.0.OR.IDEL.GT.32000) IDEL=0
      IBUF(2)=ICHAN
      IBUF(3)=IRAN
      IBUF(4)=IAV
      IBUF(5)=IDEL
      IBUF(6)=MAXMIN
```

```
IBUF(7)=ITIME
IBUF(8)=0
IBUF(9)=0
II=ISPFNW("205,MCHAN,10,IBUF(1),10)
IRES=IBUF(8)
IREM=IBUF(9)
RMLOGG=(FLOAT(IRES)+FLOAT(IREM)/FLOAT(IAV)+2048.)*(10./IRAN)
1/4096.-5./IRAN
RETURN
END
```

T A B L E 1

BUFFER BOX INTERCONNECTIONS AND A/D CHANNELS ASSIGNMENTS

B.B CH.	A/D CH.	P1-1 CONN.	COLOUR (ANALOG RETURN)
NC		1	RED (BLK)
1 AV	7 L	3	WHT (BLK)
	7 H	5	GRN (BLK)
1 DIFF	6 L	7	BLU (BLK)
	6 H	9	YEL (BLK)
1B (R-)	5 L	11	BRN (BLK)
	5 H	13	ORG (BLK)
1B (+R)	4 L	15	WHT (RED)
	4 H	17	GRN (RED)
1B (+-)	3 L	19	BLU (RED)
	3 H	21	YEL (RED)
1A (R-)	2 L	23	BRN (RED)
	2 H	25	ORG (RED)
1A (+R)	1 L	27	WHT (GRN)
	1 H	29	BLU (GRN)
1A (+-)	0 L	31	YEL (GRN)
	0 H	33	BRN (GRN)
NC		35	ORG (GRN)
NC		37	BLU (WHT)
NC		39	YEL (WHT)

B.B CH.	A/D CH.	P2-1 CONN.	COLOUR (ANALOG RETURN)
2A (+-)	8 H	1	RED (BLK)
	8 L	3	WHT (BLK)
2A (+R)	9 H	5	GRN (BLK)
	9 L	7	BLU (BLK)
2A (R-)	10 H	9	YEL (BLK)
	10 L	11	BRN (BLK)
2B (+-)	11 H	13	ORG (BLK)
	11 L	15	WHT (RED)
2B (+R)	12 H	17	GRN (RED)
	12 L	19	BLU (RED)
2B (R-)	13 H	21	YEL (RED)
	13 L	23	BRN (RED)
2 DIFF	14 H	25	ORG (RED)
	14 L	27	WHT (GRN)
2 AV	15 H	29	BLU (GRN)
	15 L	31	YEL (GRN)
3A (+-)	16 H	33	BRN (GRN)
	16 L	35	ORG (GRN)
3A (+R)	17 H	37	BLU (WHT)
	17 L	39	YEL (WHT)
3A (R-)	18 H	41	BRN (WHT)
	18 L	43	ORG (WHT)
3B (+-)	19 H	45	YEL (BLU)
	19 L	47	BRN (BLU)
NC		49	ORG (BLU)

B.B CH.	A/D CH.	P3-1 CONN.	COLOUR (ANALOG RETURN)
3B (+R)	20 H	1	RED (BLK)
	20 L	3	WHT (BLK)
3B (R-)	21 H	5	GRN (BLK)
	21 L	7	BLU (BLK)
3 DIFF	22 H	9	YEL (BLK)
	22 L	11	BRN (BLK)
3 AV	23 H	13	ORG (BLK)
	23 L	15	WHT (RED)
4A (+-)	24 H	17	GRN (RED)
	24 L	19	BLU (RED)
4A (+R)	25 H	21	YEL (RED)
	25 L	23	BRN (RED)
4A (R-)	26 H	25	ORG (RED)
	26 L	27	WHT (GRN)
4B (+-)	27 H	29	BLU (GRN)
	27 L	31	YEL (GRN)
4B (+R)	28 H	33	BRN (GRN)
	28 L	35	ORG (GRN)
4B (R-)	29 H	37	BLU (WHT)
	29 L	39	YEL (WHT)
4 DIFF	30 H	41	BRN (WHT)
	30 L	43	ORG (WHT)
4 AV	31 H	45	YEL (BLU)
	31 L	47	BRN (BLU)
NC		49	ORG (BLU)

B.B CH.	A/D CH.	P1-2 CONN.	COLOUR (ANALOG RETURN)
NC		1	RED (BLK)
5 AV	39 L	3	WHT (BLK)
	39 H	5	GRN (BLK)
5 DIFF	38 L	7	BLU (BLK)
	38 H	9	YEL (BLK)
5B (R-)	37 L	11	BRN (BLK)
	37 H	13	ORG (BLK)
5B (+R)	36 L	15	WHT (RED)
	36 H	17	GRN (RED)
5B (+-)	35 L	19	BLU (RED)
	35 H	21	YEL (RED)
5A (R-)	34 L	23	BRN (RED)
	34 H	25	ORG (RED)
5A (+R)	33 L	27	WHT (GRN)
	33 H	29	BLU (GRN)
5A (+-)	32 L	31	YEL (GRN)
	32 H	33	BRN (GRN)
NC		35	ORG (GRN)
NC		37	BLU (WHT)
NC		39	YEL (WHT)

B.B CH.	A/D CH.	P2-2 CONN.	COLOUR (ANALOG RETURN)

1A (+−)	40 H	1	RED (BLK)
	40 L	3	WHT (BLK)
1A (+R)	41 H	5	GRN (BLK)
	41 L	7	BLU (BLK)
1A (R−)	42 H	9	YEL (BLK)
	42 L	11	BRN (BLK)
1B (+−)	43 H	13	ORG (BLK)
	43 L	15	WHT (RED)
1B (+R)	44 H	17	GRN (RED)
	44 L	19	BLU (RED)
1B (R−)	45 H	21	YEL (RED)
	45 L	23	BRN (RED)
1 DIFF	46 H	25	ORG (RED)
	46 L	27	WHT (GRN)
1 AV	47 H	29	BLU (GRN)
	47 L	31	YEL (GRN)
7A (+−)	48 H	33	BRN (GRN)
	48 L	35	ORG (GRN)
7A (+R)	49 H	37	BLU (WHT)
	49 L	39	YEL (WHT)
7A (R−)	50 H	41	BRN (WHT)
	50 L	43	ORG (WHT)
7B (+−)	51 H	45	YEL (BLU)
	51 L	47	BRN (BLU)
NC		49	ORG (BLU)

B.B CH.	A/D CH.	P3-2 CONN.	COLOUR (ANALOG RETURN)
7B (+R)	52 H	1	RED (BLK)
	52 L	3	WHT (BLK)
7B (R−)	53 H	5	GRN (BLK)
	53 L	7	BLU (BLK)
7 DIFF	54 H	9	YEL (BLK)
	54 L	11	BRN (BLK)
7 AV	55 H	13	ORG (BLK)
	55 L	15	WHT (RED)
8A (+−)	56 H	17	GRN (RED)
	56 L	19	BLU (RED)
8A (+R)	57 H	21	YEL (RED)
	57 L	23	BRN (RED)
8A (R−)	58 H	25	ORG (RED)
	58 L	27	WHT (GRN)
8B (+−)	59 H	29	BLU (GRN)
	59 L	31	YEL (GRN)
8B (+R)	60 H	33	BRN (GRN)
	60 L	35	ORG (GRN)
8B (R−)	61 H	37	BLU (WHT)
	61 L	39	YEL (WHT)
8 DIFF	62 H	41	BRN (WHT)
	62 L	43	ORG (WHT)
8 AV	63 H	45	YEL (BLU)
	63 L	47	BRN (BLU)
NC		49	ORG (BLU)

CALIBRATION FACTORS FOR SHUNTS

FACTOR = NOMINAL RESISTANCE / MEASURED RESISTANCE

FOR 2 A SHUNT, NOMINAL RESISTANCE IS 25 MILLI-OHMS
FOR 10 A SHUNT, NOMINAL RESISTANCE IS 5 MILLI-OHMS

CC.	AMPS.	FACTOR
1	2	.999
2	2	1.001
3	2	.999
4	2	.998
5	10	1.000
1	10	1.000
7	10	1.002
8	10	1.002

REFERENCES

1. RT-11 Documentation Directory, Version 4, Vol. 3A, MAC, Jan. 1980. Digital Equipment Corporation, Maynard, Massachusetts.

2. Microcomputers and Memories, Chapter 4 -- Instruction Set, 1982. Digital Equipment Corporation, Maynard, Massachusetts.

3. RT-11 Documentation, Version 4, Vol. 3B, SSM, Chapter 7, March 1980. Digital Equipment Corporation, Maynard, Mass.

4. Ibid, Vol. 3A, REF, pp. 3-48 to 3-54.

APPENDIX D

THE REAL-TIME PROGRAM

A full description of the REAL-TIME program is given in this Appendix. The REAL-TIME program runs in conjunction with the WATCHDOG program (described in Appendix E) and is written in FORTRAN IV programming language for LSI-11/23 computers (1,2). The maximum number of cells that can operate simultaneously is 16. In that situation, there are 2 cells in series on each of the 8 current controllers.

D.1 CELL PARAMETERS

Each cell under test is characterized by a set of parameters which describe the cell condition. The parameters are accessed everywhere in the program: they are stored in 6 arrays, A and B (INTEGER), JT (INTEGER*4), VAL, QO and Q (REAL), in a COMMON block called PASS. A display of the arrays and their significance is given in Table 1. Index J goes from 1 to 16, corresponding to all possible cell numbers. Some parameters are automatically updated by the program, and some can be changed by the user, as described in the next section and in Appendix E.

In Table 1, it is necessary to recognize the distinction between the channels 1 to 5 and the A/D channels. The former set of numbers are assigned to a certain variable measured, i.e., 1 for the cell voltage, 2 for the positive versus reference potential, 3 for the reference versus negative potential, 4 for the current, and 5 for the temperature. In the latter case, the A/D channel numbers are related to physical connections in the system, and they range from 0 to 63, as described in Appendix C (Table 1).

TABLE 1 -- CELL PARAMETERS

A(1,J)	0 : CELL IS NOT IN TEST LOOP 1 : CELL IS IN TEST LOOP 2 : AN ERROR WAS DETECTED THAT CANNOT BE CORRECTED, TERMINATE OPERATION AND SET TO 0
A(2,J)	0 : CELL IS IN OPEN-CIRCUIT 1 : CELL IS IN ACTIVE MODE
A(3,J)	0 : CELL IS DISCHARGING 1 : CELL IS CHARGING
A(4,J)	FLAG FOR STORAGE OF DATA ON FILE (AT EVERY OTHER DATA-RECORD) 3 : DATA JUST WRITTEN TO FILE 2 : 1ST DATA-RECORD 1 : 2ND DATA-RECORD, WRITE TO FILE, RESET TO 3
A(5,J)	1 : DATA FILE OPEN FOR CELL J 0 : NO DATA FILE OPEN FOR CELL J
A(6,J)	NUMBER OF DATA-RECORDS STORED IN PRESENT BLOCK (SEE A(7,J))
A(7,J)	BLOCK NUMBER TO BE WRITTEN ON DATA FILE
A(8,J)	FLAG FOR DATA-TAKING RATE ADJUSTMENT ROUTINE 0 : START OF CHARGE OR DISCHARGE > 0 : OTHERWISE
A(9,J)	CURRENT CONTROLLER ASSOCIATED WITH CELL J
A(10,J)	FULL SCALE CURRENT VALUE FOR 50 MV SHUNT(IN A)
A(11,J)	CELL WITH WHICH CONNECTED, 0 IF NONE
A(12,J)	NUMBER OF A/D CHANNELS ACTIVE
A(13,J)	CHANNEL 1(CELL VOLTAGE) : CORRESPONDING A/D CHANNEL NUMBER
A(14,J)	CHANNEL 1 : A/D RANGE
A(15,J)	CHANNEL 2(+REF) : IDEM
A(16,J)	CHANNEL 2 : IDEM
A(17,J)	CHANNEL 3(REF-) : IDEM
A(18,J)	CHANNEL 3 : IDEM
A(19,J)	CHANNEL 4(CURRENT) : IDEM
A(20,J)	CHANNEL 4 : IDEM
A(21,J)	CHANNEL 5(TEMPERATURE) : IDEM
A(22,J)	CHANNEL 5 : IDEM
B(1,J)	CHANNEL(1-5) WATCHED FOR REGULATION OF DATA-TAKING RATE ON DISCHARGE
B(2,J)	CHANNEL(1-5) WATCHED FOR REGULATION OF DATA-TAKING RATE ON CHARGE
B(3,J)	DISCHARGE MODE 1 :CONSTANT CURRENT 2 :IDEM, 1 HALF-CYCLE 3 :IDEM, 2 CELLS IN SERIES 4 :CONSTANT POWER 5 :IDEM, 1 HALF-CYCLE 6 :IDEM, 2 CELLS IN SERIES 7 :PULSED CURRENT 8 :IDEM, 1 HALF-CYCLE 9 :IDEM, 2 CELLS IN SERIES

B(4,J) CHARGE MODE
1 : CONSTANT CURRENT
2 : IDEM, 1 HALF-CYCLE
3 : IDEM, 2 CELLS IN SERIES
4 : CONSTANT VOLTAGE - LIMITED CURRENT
5 : IDEM, 1 HALF-CYCLE
6 : IDEM, 2 CELLS IN SERIES
7 : PULSED CURRENT
8 : IDEM, 1 HALF-CYCLE
9 : IDEM, 2 CELLS IN SERIES

B(5,J), B(6,J), B(7,J) } CHANNELS(1-5) PROGRAMMED FOR LIMITS ON DS.

+CHANNEL IS UPPER LIMIT, -CHANNEL LOWER LIMIT
E.G +1 IS UPPER LIMIT ON CELL VOLTAGE
-3 IS LOWER LIMIT ON REF-

B(8,J), B(9,J), B(10,J) } CHANNELS(1-5) PROGRAMMED FOR LIMITS ON CH.

IDEM

B(11,J) NOT SPECIFIED

B(12,J) NOT SPECIFIED
USED TO STORE B(2,J) WHEN OPERATION IN CVLC-MODE (B(2,J) IS THEN 4)

B(13,J) VOLTAGE CHANNEL(1-3) FOR CONST. POWER DS.

B(14,J) CONTROLLED VOLTAGE CHANNEL(1-3) FOR CVLC-MODE

JT(1,J) TIME FOR TAKING NEXT DATA

JT(2,J) TIME FOR NEXT CURRENT CONTROL

JT(3,J) TIME FOR COMPLETION OF PRESENT STATE

JT(4,J) TIME INTERVAL BETWEEN SUCCESSIVE STORAGE OF DATA-RECORDS

JT(5,J) DURATION OF CHARGE

JT(6,J) DURATION OF DISCHARGE

JT(7,J) DURATION OF OPEN-CIRCUIT

JT(8,J) TIME INTERVAL BETWEEN SUCCESSIVE PASSES THROUGH THE CURRENT CONTROL ROUTINE

JT(9,J) TIME FOR RESETTING THE CONTROLLER DISCONNECT TIMER

JT(10,J) TIME INTERVAL BETWEEN SUCCESSIVE RUNS THROUGH THE DATA-TAKING ROUTINE

VAL(1,J) INTERVAL THAT REGULATES THE DATA-TAKING RATE ON DISCHARGE

VAL(2,J) INTERVAL THAT REGULATES THE DATA-TAKING RATE ON CHARGE

VAL(3,J) CALIBRATION VALUE FOR CURRENT CONTROLLER J

VAL(4,J) CHARGE EXCESS FACTOR

VAL(5,J), VAL(6,J), VAL(7,J), VAL(8,J) } LIMITS ON DISCHARGE

VAL(9,J), VAL(10,J) } LIMITS ON CHARGE

VAL(11,J) CURRENT ON DISCHARGE (IF CONTROLLED)

VAL(12,J)	CURRENT ON CHARGE (IF CONTROLLED)
VAL(13,J)	POWER VALUE FOR CONSTANT POWER DISCHARGE
VAL(14,J)	VOLTAGE VALUE FOR CVLC-MODE
QO(1,J)	MAXIMUM A-HRS TO BE PASSED ON DISCHARGE
QO(2,J)	MAXIMUM A-HRS TO BE PASSED ON CHARGE
Q(1,J)	A-HRS PASSED ON DISCHARGE
Q(2,J)	A-HRS PASSED ON CHARGE

D.2 PARAMETER FILES

The program works with 2 files that contain information about the parameters described in the previous section. The names of these files are FMAT.UNF and GMAT.UNF, and they are of similar format. Their purpose is to provide a procedure to place a cell in operation. There are 2 ways in which one can start a test: (See also Sections 2.4.2.1 and E.2.7.(RUN))

1. The "initialize" mode:
    1. A new cell goes under test, at cycle 1.
    2. A cell that was running at cycle n is started at cycle n+1, but with a different cycling mode.
2. The "re-start" mode : a cell is started exactly where it left off:
    1. If the cell was interrupted on a charge, continue the charge.
    2. If the cell was interrupted at the end of a charge, start a discharge.
    3. Etc...

FMAT.UNF and GMAT.UNF are used in the following way:

1. When a user wants to place a new cell under test, or completely respecify the way a cell is operating, he must run the INITIALIZATION section of the WATCHDOG program (see Appendix E). The answers to all questions asked by this routine are stored in the appropriate slot of FMAT.UNF, for the appropriate cell number. The user can do this at any time, even when a cell with that same number is running. The parameters are stored in FMAT.UNF, but no action is taken by the

REAL-TIME program. When the user operates the RUN section of the WATCHDOG program (Appendix E), in the "initialize" mode, the REAL-TIME program opens FMAT.UNF, gets the information for the cell, stores it in the 6 parameter arrays, puts the cell in the test loop, and closes FMAT.UNF.

2. The REAL-TIME program regularly updates GMAT.UNF, with the cell parameters modified during operation. For each cell, it also stores the date, time of day and cycle number. If a cell's operation is stopped, either through a normal stop request or through a program crash, GMAT.UNF contains a last update of all the information needed to immediately re-start the cell. To implement, the user must run the RUN section of the WATCHDOG program in the "re-start" mode. the REAL-TIME program keeps GMAT.UNF open at all times, so it just retrieves the information for that cell and puts it in the parameter arrays.

3. Cell parameters can be changed selectively, by running the CHANGE section of the WATCHDOG program, and GMAT.UNF is automatically updated.

A diagram of the structure of FMAT.UNF and GMAT.UNF is shown in Table 2. The files are defined with a record size equal to 1 real number. The (*) mark means that the information is defined only in GMAT.UNF and not in FMAT.UNF. JMAX is the maximum number of cells, which is set to 16. ICYCLE(J) is the cycle number at which cell J is running. If cycle numbers for cell J are updated at the beginning of a charge, MCYCLE(J)=1; if at the beginning of discharge, MCYCLE(J)=0.

PARAMETER FILE STRUCTURE

RECORD	STORAGE	CELL
1	JMAX	
2	DATE DAY,MONTH(*)	1
3	DATE YEAR, -- (*)	1
4	TIME(*)	1
5	A(1,1),A(2,1)	1
.	.	
.	.	
15	A(21,1).A(22,1)	1
16	B(1,1),B(2,1)	1
.	.	
.	.	
22	B(13,1);B(14,1)	1
23	JT(1,1)	1
.	.	
.	.	
32	JT(10,1)	1
33	VAL(1,1)	1
.	.	
.	.	
46	VAL(14,1)	1
47	Q0(1,1)	1
48	Q0(2,1)	1
49	Q(1,1)	1
50	Q(2,1)	1
51	DATE DAY,MONTH(*)	2
.	.	
.	.	
2+(J-1)*49	DATE DAY,MONTH(*)	J
.	.	
.	.	
50+(J-1)*49	Q(2,J)	J
.	.	
.	.	
785	Q(2,16)	16
786	ICYCLE(1),MCYCLE(1) (*)	1
787	ICYCLE(2),MCYCLE(2) (*)	2
.	.	
.	.	
801	ICYCLE(16),MCYCLE(16) (*)	16

D.3 DATA-FILE STRUCTURE

Each cell under test stores data in its own scratch data file. The data file for cell J (J ranging from 1 to 9) is called DATOJO.UNF, for cell JJ (JJ from 10 to 16), DATJJO.UNF. Table 3 shows the structure of a block on a scratch-file. A block is 256 words long, which is equal to 128 real words. The last 8 real words serve as a pointer and A-hr storage area. Therefore, there are only 120 real words available to store data. In this example, at each measuring time, 4 A/D channels were measured (VCEL,+REF,REF- CURR) and the results stored. Therefore, each data-record is 5 real words long, and the maximum number of data-records in each block is 120/5 = 24.

TABLE 3 -- DATA FILE STRUCTURE

0							
		[MONTH]	[DAY]	[YEAR]	[CYCLE]	(e.g. start of charge)	
4							
				[TIME]		[VCEL]	
8							
[+REF]		[REF-]		[CURR]		[TIME]	
12							
				.			
				.			
				.			
				.			
		[MONTH]	[DAY]	[YEAR]	[CYCLE]	(e.g. start of discharge)	
X							
X+4							
[+REF]		[REF-]		[CURR]		[TIME]	
X+8							
				.			
				.			
				.			
				.			
[CH POINTER]		[OC POINTER]		[DS POINTER]		[NSTOP]	
120							
[A-HR DS]		[A-HR CH]		[INTERR]			
124							

A date specification and a cycle number are stored as the 1st record of each charge or discharge (not open-circuit). Charge, discharge, or open-circuit data may not necessarily start in the beginning of a block. Pointers indicate where in the block the three types of data start. If one or two types of data are not stored on that block, their respective pointers are 0. Pointers are defined in terms of number of data-records (here: 5 real words each) since the beginning of the block. The date specification is counted as a data-record.

This type of storage does not allow the same data-type to be stored twice in different parts of a block (e.g. open-circuit data after charge and after discharge). In that case, a block is left only partially filled, and data storage continues on the next block. A pointer is used, NSTOP, that points to the last data-record stored on the block.

NSTOP = 0 when block full, and data continue without interruption on next block

> 0 when data-taking interrupted at that record (non-matching times with eventual data on next block)

< 0 when block is not completely full and data continue without interruption on next block

= 300 when no valid data in the block

At certain times, a cell's operation needs to be interrupted: when that cell is put back in operation, data-storage starts at a new block. However, the program that decodes data must have some information indicating that there was an interruption between the end of a block and the start of the subsequent one, and that the times are not matched. A

flag, INTERR, is provided for that purpose.

INTERR = 1 when data-taking has been interrupted at the end of the block, and resumed (non-matching times) with the start of the next block

INTERR = 0 in all other cases

A list of all the possible combinations of NSTOP and INTERR is given below:

1) NSTOP=0, INTERR=0

   data-block is full, continues on next block, matching times

2) NSTOP=I, INTERR=0

   I data-records in this block, last block of data

3) NSTOP=-I, INTERR=0

   I data-records in this block, continues on next block, matching times

4) NSTOP=I, INTERR=1

   I data-records in this block, continues on next block, non-matching times

5) NSTOP=-I, INTERR=1

   illegal

6) NSTOP=300, INTERR=0

   invalid block, no more data

7) NSTOP=300, INTERR=1

   invalid block, continues on next block, non-matching times

8) NSTOP=0, INTERR=1

   illegal

D.4 THE REAL-TIME LOOP

The REAL-TIME loop is shown in Figure 1.

The action timers and action time intervals are stored in the JT array, described in Section D.1. Further details are given in the description of the subroutines.

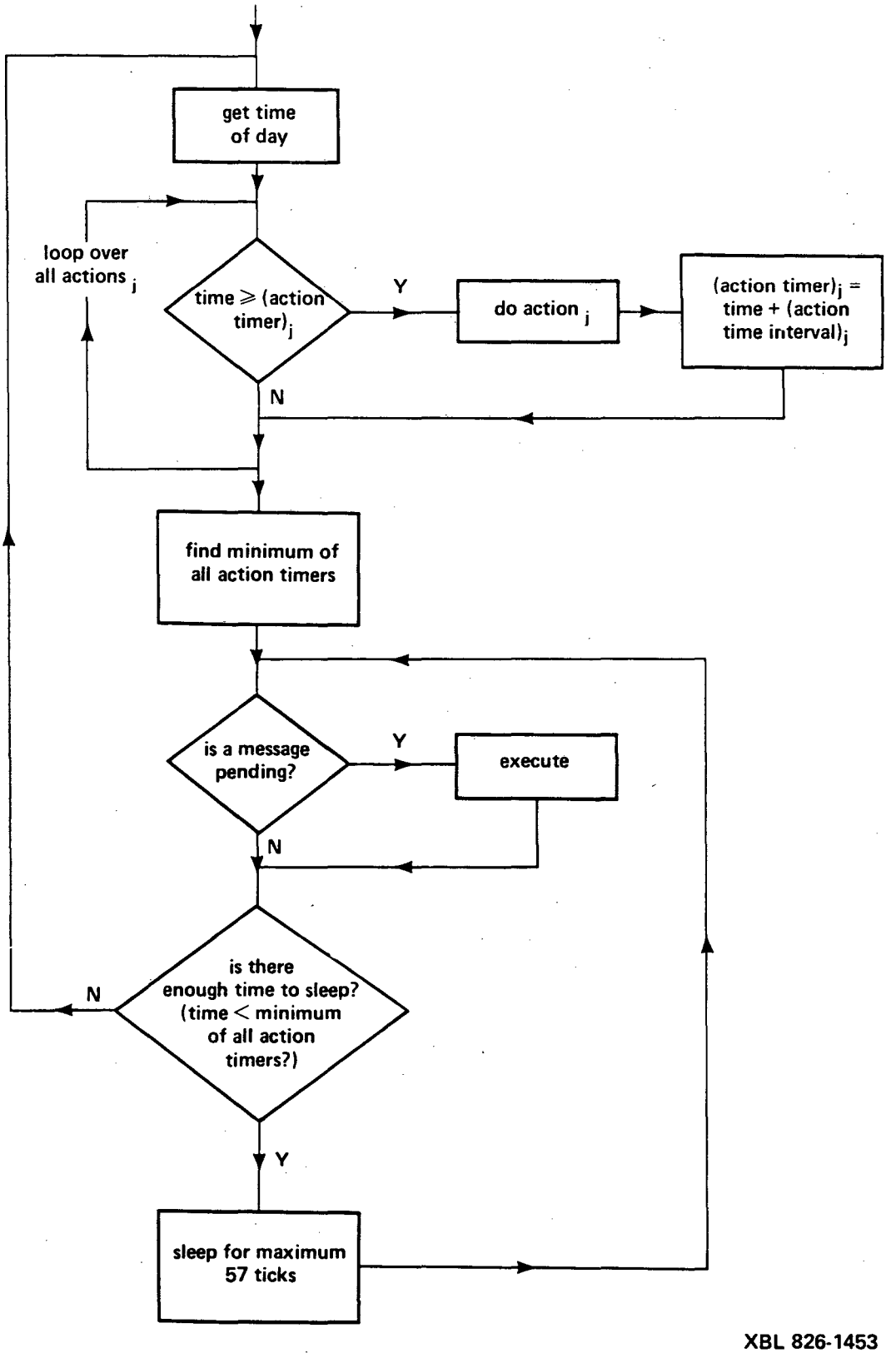


FIGURE 1. The Real-Time Loop.

D.5 THE MESSAGE EXCHANGE PROCEDURE

Chapter 2, Section 2.4.2.2, gives a list of the procedures a user can initiate by communicating with the REAL-TIME program. This communication is implemented through an organized message exchange between the REAL-TIME program (in the foreground) and the WATCHDOG program (in the background). It is always the WATCHDOG program which starts an exchange. It sends a message to the REAL-TIME program, and the latter returns a response. This is illustrated in Fig. 2.

The WATCHDOG program sends a message, and it waits until it receives a response. The REAL-TIME program sends a response, but it does not wait for the WATCHDOG program to send another message. This scenario respects the more time-critical control operations performed by the REAL-TIME program. It works with an 'automatic' message receiver, which is a short routine that interrupts the normal program execution (REAL-TIME loop) when it senses that there is a message from the WATCHDOG program. The receiver sets a message flag and then once a second, the REAL-TIME program transfers control to a message handler routine, which determines if the message flag is set. If that is the case, the handler organizes the execution of a task to respond to the particular message.

For some operations, one message exchange is required, for others, two or three. At various times, an 'abort the procedure' message can be sent to the REAL-TIME program. Fig. 3. illustrates a typical 2-pass message exchange operation.

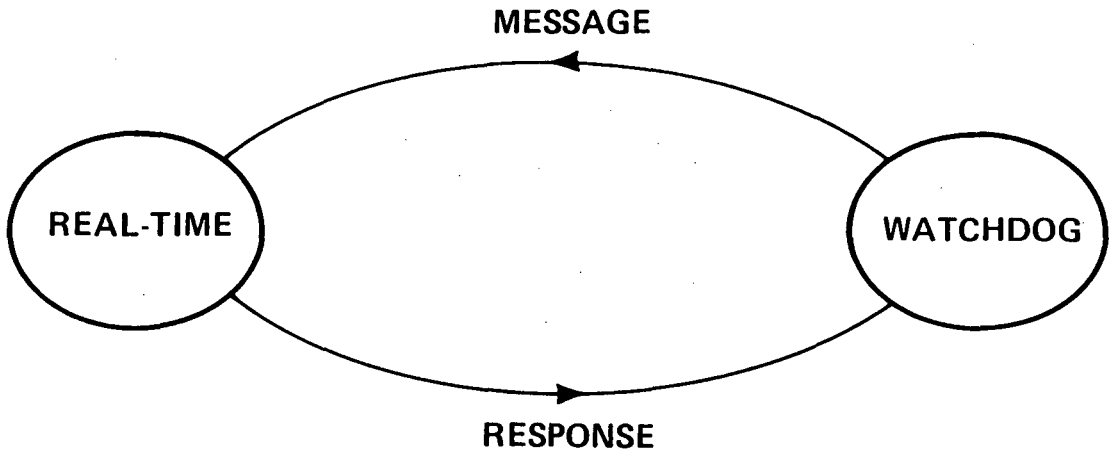


FIGURE 2. Flow of Information Between the REAL-TIME and WATCHDOG Programs.

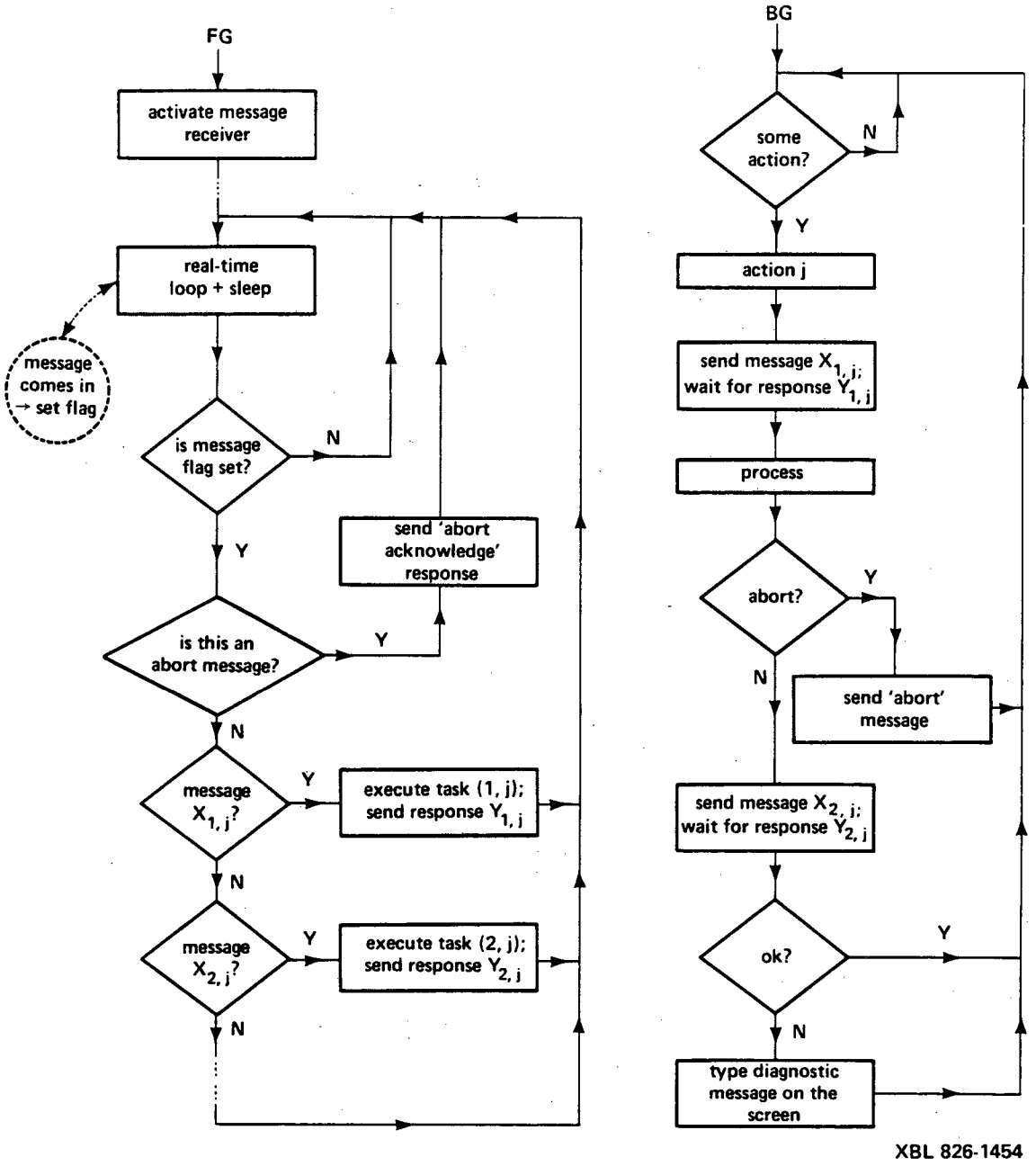
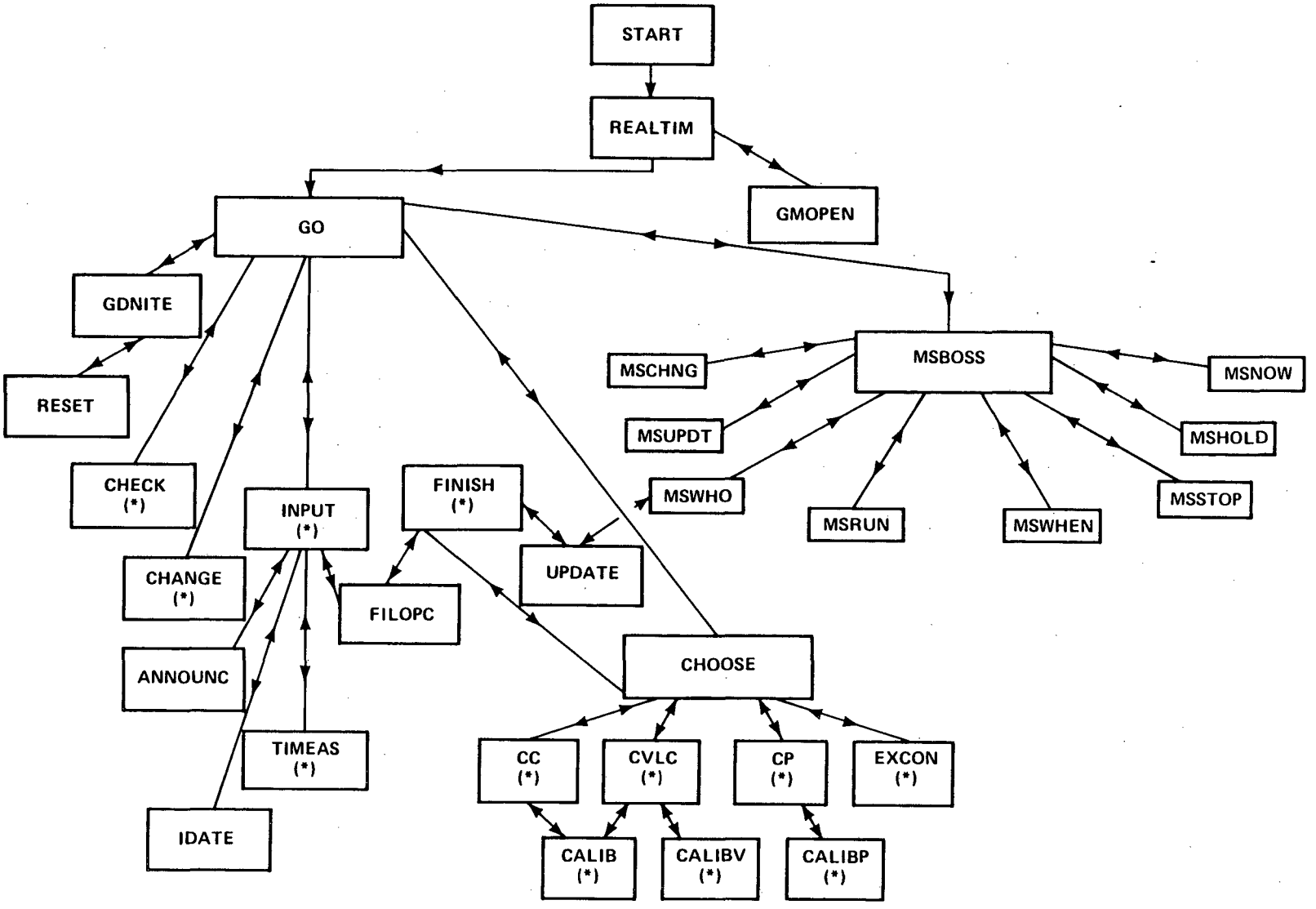


FIGURE 3. Protocol for a Two-pass Message Exchange between the REAL-TIME (Foreground) and the WATCHDOG (Background) Programs.

D.6 SUBROUTINE DIAGRAM

A schematic diagram of the subroutines of the REAL-TIME program is given in Fig. 4. Arrows pointing down are for calls to subroutines, and arrows pointing up for return from subroutines. Two completion routines (CRMESS and CRSTOP) are not in the diagram, and they are not called from any of the other subroutines. Another subroutine, HDCONC, is not shown on the diagram. It contains all the I/O commands, and is called from various subroutines. The subroutines that call HDCONC are marked with a (*).



XBL 826-1450

FIGURE 4. Subroutine Diagram of the REAL-TIME Program.

Arrows pointing down are calls to subroutines, arrows pointing up are returns from subroutines; asterisks indicate subroutines which call the I/O library; completion routines CRMESS and CRSTOP are not included in the figure.

D.7 SUBROUTINE DESCRIPTIONS

The order in which the subroutines are described is the same as the order in which their code is given, as shown in section D.10.

D.7.1 START

START is a short routine that calls the main program. It has no other purpose than to allow the REAL-TIME program to run as a virtual job (explained in Appendix B).

D.7.2 REALTIM

REALTIM initializes certain variables and opens a communication path with the HD I/O driver.

D.7.3 GMOPEN

GMOPEN opens FMAT.UNF and GMAT.UNF. It first reads JMAX, which is the maximum number of cells in the loop, then closes FMAT.UNF.

D.7.4 GO

GO is the routine which implements the REAL-TIME loop. Its initial section defines some timers and flags, activates the message receiver completion routine, and activates a ^C^C intercept function. Then it enters the REAL-TIME loop, from which it never leaves unless a ^C^C is typed.

The first part of the REAL-TIME loop consists of a series of DO-loops, each executed for all cells. The clock time is compared to the action timers (see Section D.4) and if necessary, control is transferred

to specific routines.

1. The hardware check (CHECK) and parameter update (UPDATE) loop. If an error is detected (A(1,J)=2), a last measurement is taken (INPUT) and the cell's operation is terminated (FINISH).

2. The status change loop (charge, open-circuit, discharge). This is done by repeated calls to various sections of the CHANGE routine, with intermittent calls to other routines (CHOOSE, INPUT). If a stop flag is set (ISTOP), the cell's operation is terminated (FINISH). The loop for the status change of a cell is executed twice for all cells (necessary for cells that run in series). For example, if cell 15 is connected with 3, and at the first pass 15 needs a status change, 3 will go through the same status change at the second pass.

3. The current control loop (CHOOSE). If a stop flag is set (ISTOP), control is immediately transferred to the status change loop.

4. The data taking loop (INPUT). If a limit flag is set (IEX) control is immediately transferred to the status change loop.

- - - - -

The second part of the REAL-TIME loop consists of the following operations:

1. A call to GDNITE, to define the minimum of all action timers for all cells.

2. A check for a ^C^C entered (IFLAG not = 0). In that case, a last measurement is taken for all cells (INPUT), and then the REAL-TIME loop is terminated by a call to FINISH.

3. A call to MSBOSS, which accommodates all message exchanges. If the no-sleep flag is set (INITE), control is immediately transferred to the start of the REAL-TIME loop.

4. A check, on a termination flag (A(1,J)=2), which might have been set in various places in the program. In that case, a last measurement is taken (INPUT) and the cell's operation is terminated (FINISH).

5. A call to GDNITE, which executes a sleep command.

* D.7.5 GDNITE

GDNITE has two sections, which execute according to the value of the input parameter IGD.

The first section finds the minimum of all action timers. The second section checks if the time is 23:59:55, in which case it calls RESET. It then checks if there is enough time to sleep, in which case it sleeps for any time smaller or equal to 57 ticks. If there is no time to sleep, it sets the no-sleep flag (INITE) and exits.

D.7.6 ANNOUNC

ANNOUNC types a time on the screen, in hrs,mins,secs,ticks.

D.7.7 IDATE

IDATE returns the current date.

D.7.8 CHANGE

CHANGE implements the CHANGE of status of a cell (from open-circuit to charge or discharge, and vice-versa). It runs in parallel with the status change loop of GO, with the different entry- and exit points associated to values of IIN and IOUT. CHANGE connects or disconnects a current controller. At the start of a charge or discharge, it measures the offset of the current controller. CHANGE checks whether or not data storage can continue on the same block in the data-file (see section D.3). It looks at the pointer for the mode it is entering, located in the BUF storage area (see INPUT). That pointer is at position 120 + M, with M equal to 1 for charge, 2 for open-circuit, 3 for discharge. If the pointer value is 0, data for that mode can continue on the block.

D.7.9 CHECK

This routine checks if a controller is in computer mode and if the current has the right polarity (+ for discharge, - for charge). If an error is detected, a termination flag is set (A(1,J)=2).

D.7.10 CHOOSE

CHOOSE transfers control to one of the current control routines, according to the charge or discharge mode of the cell.

D.7.11 CC

CC implements the "constant current" charge or discharge; it is divided in three sections. In the initialization section, it defines the maximum number of A-hrs to pass on charge (if it is in that mode). In the control section, it updates the A-hrs and determines if they do not exceed the limit, and calls CALIB. In the termination section, it updates the A-hrs of that mode and initializes the A-hrs of the next mode.

CC also takes care of 2 cells cycled in series: the cell with the highest number is the one on which the controls are performed (i.e. the one for which CALIB is called).

The last command pertains to the case of a charge mode that gets interrupted: an imaginary number of discharged A-hrs is defined, so when the cell is restarted, the proper number of A-hrs to pass on charge is computed in the initialization section of this routine.

D.7.12 CVLC

CVLC implements the "constant voltage - limited current" charge; it is divided into three sections, which have the same basic functions as in CC. In the control section, it decides on whether the control mode should be constant current or constant voltage (if the voltage limit has been reached), and calls CALIB or CALIBV. In the case of constant voltage control, the channel to be watched for the data-storage rate adjustment is set to channel 4 (current), and the adjustment interval is 20 mA (see TIMEAS).

CVLC also takes care of 2 cells cycled in series; for the constant current part, the cell on which the controls are performed is the cell with the highest number (i.e., the one for which CALIB is called). For the constant voltage part, it is the cell with the highest voltage. (i.e., the one for which CALIBV is called).

In CVLC, A(8,J) has a special significance (see Section D.1):

1. A(8,J) = 0 : start of a charge (or a discharge).

2. A(8,J) = 1 : set by TIMEAS after the 1st pass. For all the other control modes, it stays at that value, but for CVLC, it is further modified.

3. A(8,J) = 2 : when current control is performed on the cell.

4. A(8,J) = 4 : when current or voltage control is performed on the companion cell.

5. A(8,J) = 3 : when voltage control is performed on the cell.

When voltage control is performed on a cell, and it has a companion cell, control can switch to the other cell, if it exceeds its own imposed voltage limit.

The last command in CVLC serves the same function as in CC.

D.7.13 CP

CP implements the "constant power" discharge; it is divided into 3 sections, which have the same basic functions as in the CC and CVLC routines. The control section calls CALIBP. At the first pass, CP puts a

.2A charge current in the cell, for 3 ticks, then a .2A discharge current, for the same time. This provides a starting point for the power regulation in CALIBP.

CP also takes care of 2 cells cycled in series; it is the cell with the highest voltage on which the controls are performed, (i.e., the one for which CALIBV is called).

D.7.14 EXCON

EXCON implements the "pulsed current" charge or discharge; it is divided into 3 sections, which have the same basic functions as in the CC, CVLC and CP routines. The control section does not call any other routine, because the current is not regulated, rather, it is imposed by an external voltage profile. A few important points concerning this mode are:

1. In the initialization section, the controller is set in the external mode.

2. The cell voltage (channel 1), and eventually the reference voltages (channels 2 - 3), are measured at the OFF-side of the pulse. The cell voltage is also measured at the ON-side of the pulse, as channel 5 (which is usually the temperature channel).

3. The current (channel 4) is not measured with its usual A/D channel. The program automatically switches to an A/D channel that is wired to an averager circuit. In the Buffer Box interconnection diagram (see Appendix C, Table 1.), this A/D

channel number is always 1 higher than the usual A/D channel for the current measurement.

4. A-hrs are not calculated until after 2.5 minutes from the beginning of the charge or discharge; this allows the current averager circuit to come to a stable value.

EXCON also takes care of 2 cells cycled in series; the cell with the highest cell number is the one on which the controls are performed. The last command serves the same function as in CC and CVLC. For additional information on the pulsating mode, see RDLOGG. (Section D.7.36)

D.7.15 CALIB

CALIB implements the constant current control. The algorithm is

$$(\text{I set})_{i+1} = (\text{I set})_i * \text{setpoint}/(\text{I meas.})_i$$

The setpoint and the measured current are in volts. No conversion is made to amperes.

A parameter ANS is returned as Y, N or F. Y means that regulation within the minimum specified accuracy range was obtained in at most 10 iterations. N means that regulation within the maximum specified accuracy range could not be obtained in 10 iterations. F means that something is really wrong, and that the cell should be taken out of operation.

D.7.16 CALIBV

CALIBV implements the constant voltage control. The algorithm is

$$(I\ set)_{i+1} = (I\ set)_i + (setpoint - V_i)/X_i$$

with $X_i = (V_i - V_{i-1})/((I\ meas.)_i - (I\ meas.)_{i-1})$

The setpoint and the measured voltage are in volts, the measured current is converted to amperes. The parameter ANS is used in the same way as for CALIB.

D.7.17 CALIBP

CALIBP implements the constant power control. The control algorithm is

$$(I\ set)_{i+1} = (I\ set)_i + (setpoint - P_i)/(V_i + X_i * (I\ meas.)_i)$$

with $P_i = V_i * (I\ meas.)_i$

and $X_i = (V_i - V_{i-1})/((I\ meas.)_i - (I\ meas.)_{i-1})$

The setpoint is in watts, the measured voltage in volts, and the measured current is converted to amperes. The parameter ANS is used in the same way as for CALIB and CALIBV.

D.7.18 INPUT

INPUT regulates data taking, storage and limit checking. The structure of a block on a scratch data-file is explained in Section D.3. Each cell under test uses a 128 real word storage area, BUF(I,J) (with J = cell number and I = 1 to 128), and a pointer, A(6,J), to the last data-record stored in the area. The last 8 real words of the BUF area are used for special pointers and for A-hr storage; the same 8 words are found in each block of the scratch-files, as explained in Section D.3. In the same section, it is explained that a data-record consists of a

time and a number of real words: for example, for 4 measurements (cell voltage, positive vs. reference, reference vs. negative, current), a data-record is 5 real words long and the maximum number of records that fit in a block is 24. When a new data-record is added to the storage area, the pointer A(6,J) is increased by 1. At every other data record that is stored for the cell, the 128 real word storage area is written out to the scratch data file, at the current block number (A(7,J)). Therefore, the block on the file is gradually filling with data. When it is full, or for some other reason (see section D.3), a new block has to be used, the last 8 real words of the BUF storage area are reset to 0, A(6,J) is reset to 0, the block number (A(7,J)) is increased by 1, and the procedure re-starts.

INPUT is called in 4 different modes, associated to the value of the input parameter IMODE.

1. IMODE = 0 : this is the normal mode. INPUT calls TIMEAS, which decides whether or not data must be stored in the BUF storage area (ANS = Y or N). Then, INPUT takes data, checks limits (sets IEX=1 if a limit is exceeded), eventually stores data (depending on ANS) or writes the storage area to the file.

2. IMODE = 1 : INPUT does not take data, store data, or check for limits. It writes the storage area to the file and re-initializes for a new block.

3. IMODE = 2 : INPUT takes and stores data, and eventually writes the storage area to the file, etc.

4. IMODE = 3 : INPUT calls IDATE and stores the date and cycle number in the BUF storage area, eventually writes it to the file but does not take data or check limits.

D.7.19 TIMEAS

TIMEAS implements the adjustment of the rate of data storage. It is called by INPUT (only when INPUT runs in mode 0) and returns a flag, ANS. ANS = Y when it is time to store a data-record in the BUF storage area. ANS = N when a data-record must only be checked for limits, then discarded. Two action time intervals are connected with subroutines INPUT and TIMEAS : ΔTinp and ΔTstore, or JT(10,J) and JT(4,J), respectively (defined in Section D.1). ΔTinp is the time interval between successive runs through INPUT and TIMEAS, i.e., the time interval between successive checks for limits (INPUT), or successive adjustment of the data-storage rate (TIMEAS). ΔTstore is the actual time between the storage of 2 successive data records for a cell. ΔTstore is originally specified by the user, with a maximum of 16 minutes and a minimum of 2 minutes. The user also assigns a channel (most often the cell voltage, channel 1) to be followed for data-storage rate adjustment. For example, the cell voltage is known to vary rapidly at the beginning and end of a charge or discharge, but to have a steady plateau value in the middle; that is the reason for a rate adjustment.

ΔTstore, as specified by the user, is the storage time interval during the period where the channel followed is at its plateau value. When the measurements on that channel start to exhibit variation, ΔTstore is reduced by the program. In parallel, ΔTinp is reduced, because adjustment of the rate needs to happen more frequently. When the variation

decreases, ΔTstore and ΔTinp are increased. ΔTinp is 15 seconds, 30 seconds, or 1 minute. The corresponding values of ΔTinp and ΔTstore are given in the following table.

ΔTinp	ΔTstore (FOR 2 MIN. PLATEAU RATE)	ΔTstore (FOR 16 MIN. PLATEAU RATE)
15 SECS	30 SECS	30 SECS
30 SECS	1 MIN	APP 3 MIN
1 MIN	2 MIN	16 MIN

For other plateau rates, the intermediate times are between 1 and 3 minutes. The factor that is used to cut or multiply ΔTstore is called JFAC(J) (J = 1 to 16). Its value depends on the value of ΔTstore specified by the user (ΔTstore,p). When ΔTstore,p is cut twice, it is equal to 30 seconds.

JFAC=2**X

with X=[log(2*ΔTstore,p)]/2*log 2

this satisfies

ΔTstore,p/[(2**X)**2]=.5 min

JFAC(J) is computed in routine MSRUN.

D.7.20 FILOPC

FILOPC opens (IOP not = 0) or closes (ICLO not = 0) a file. It encodes a file name from the number of the cell for which the operation is carried out. To open a file, it first tries a LOOKUP command. If

the file does not exist, an error is returned, upon which it tries an IENTER command and writes block 299 of the file, to define its length.

D.7.21 UPDATE

This routine updates GMAT.UNF. It is called from 4 different places, corresponding to a value of the input parameter IMOD. In each case, cell parameters that were modified are written to GMAT.UNF.

1. IMOD = 1 : called from the status change loop, in GO. The cell is in a new mode.

2. IMOD = 2 : called from FINISH. The cell goes out of operation.

3. IMOD = 3 : called from the hardware check loop, in GO : time, date and A-hrs are updated.

4. IMOD = 4 : called from MSUPDT : a large part of the parameter arrays is updated.

A special condition is that of the single half-cycle (charge or discharge), with a time limit, rather than an A-hr limit, to terminate the mode. When IMOD = 2 or 3, the time of day is subtracted from the termination time, to give the remaining operation time.

At the end of UPDATE, the 1st and last record of GMAT.UNF are written into; this guarantees that all the information is effectively written into the file.

D.7.22 FINISH

FINISH terminates the operation of one (IALL = 0) or all (IALL = 1) cells. A termination of all cells happens when a ^C^C is typed on the foreground terminal; in that case, execution stops in FINISH. If only one cell is put out of operation (due to an error, a STOP or HOLD request), control returns to the real-time loop.

D.7.23 RESET

RESET takes care of the 'midnight roll-over'. At midnight, the time switches from 23:59:59:59 to 0. RESET is called from GDNITE, 5 seconds before midnight. It subtracts 24 hours from all timers, waits until midnight has passed, then returns.

d.7.24 CRMESS

CRMESS is the message receiver. It is a completion routine which is called when a message comes in from the background. It increments a flag, MESCNT.

D.7.25 MSBOSS

MSBOSS is the message handler: it organizes the message exchanges. Messages that come in from the WATCHDOG program are automatically put in a global INTEGER array, MSG, with dimension 59. Responses that go to the WATCHDOG program are stored in a local INTEGER array (in various subroutines), RESP. The first word of a message that comes in is always the number of INTEGERs transferred (RT-11 specification). In other words, the first word that the sending program puts in its message is to be found at the second position in the message that the receiving

BD: the operation has been aborted, because MSG(1) was BD or nothing else the program could recognize

TH: the user wants to start a cell with a number higher than JMAX -- abort

NG: for RUN the user wants to start a cell which is already in operation -- abort

for STOP or HOLD : there is already a STOP request pending for the cell -- abort and cancel the STOP request

SC: the cell cannot be started on its own, it has a companion cell (connected in series) -- wait

NC: the user wants to stop a cell which is not running -- abort

EF: error in opening FMAT.UNF, in RUN routine -- abort

MSBOSS decodes the first word of an incoming message and calls the appropriate message routine. There are two cases where MSBOSS calls a message routine (but not upon receival of a message):

1. When a cell goes in open-circuit, CHANGE sets a flag, IHOLD. In that case, MSBOSS automatically calls MSHOLD, to check if any cell must be interrupted at the end of a half-cycle.

2. When a timer request for the STOP operation expires, the completion routine CRSTOP is called, which sets the flag IMSTOP. In that case, MSBOSS calls MSNOW.

program gets. This is explained in the RT-11 documentation (1).

In the procedure adopted here, MSG(2) always contains a code identifying the type of operation that is being implemented, and MSG(3) always contains the cell number. The operations are:

1. Change the parameters of a cell (CHANGE);
2. Display the parameters of a cell (DISPLAY);
3. Stop a cell in time ΔT (STOP);
4. Stop a cell after a half-cycle to come (HOLD);
5. Start a cell (RUN).

The codes are

CH : 1st pass of the CHANGE operation

DS : only pass of the DISPLAY operation

T1 : 2nd pass of the CHANGE operation

ST : 1st pass of the STOP operation

HD : 1st pass of the HOLD operation

T2 : 2nd pass of the STOP operation

T3 : 2nd pass of the HOLD operation

RR : 1st pass of the RUN operation

T4 : 2nd and 3RD pass of the RUN operation

BD : bad, abort operation

The 1st word of RESP is a code which informs the WATCHDOG program of any problems in the message exchange procedure. The codes are:

OK: everything is fine, the transfer is completed

The message routines cannot fully be understood without reading the corresponding parts of the WATCHDOG program. The message data-structures are not described; they are apparent from a comparison of the corresponding parts of the REAL-TIME and WATCHDOG programs.

D.7.26 MSWHO

MSWHO is called at the first pass of the RUN operation. It returns information to the WATCHDOG program, about the status of the cell and the time and date when it was last in operation.

D.7.27 MSRUN

MSRUN is called at the second pass of the RUN operation. Depending on the mode, 'initialize' or 're-start', it reads parameters from FMAT.UNF or GMAT.UNF into the cell parameter arrays. If the user wants a new scratch-file (ISCRAT = 0), A(7,J) is set to 0. Otherwise, the INTERR flag (see Section D.3) is set in the last block where data were stored (before interruption), and A(7,J) is incremented. MSRUN initializes other parameters, such as JFAC(J) (see TIMEAS). If it senses that a cell is in series with the cell that the user wants to run, it returns the 'SC' response, and waits to be re-called for the companion cell, so it can start the two cells together.

D.7.28 MSCHNG

MSCHNG is called at the first pass of the CHANGE operation, or the only pass of the DISPLAY operation. It sends the contents of the parameter arrays and the cycle number to the WATCHDOG program. If the cell is not running, it first gets that information from GMAT.UNF.

D.7.29 MSUPDT

MSUPDT is called at the second pass of the CHANGE operation. It updates some parameters in the parameter arrays, and calls UPDATE to write them into GMAT.UNF. Some special cases are:

1. If the cell is in operation, the A(I,J) parameters are never updated.

2. If MSG(32) and MSG(33) are zero, then JT(4,J) (= ΔTstore,p, see TIMEAS routine) was not changed. If they are not zero, they contain the modified value of JFAC(J) (see TIMEAS routine). Then, JT(4,J) is immediately written out to GMAT.UNF. However, its value for the program has to be modified, so that it corresponds to the value of JT(10,J) (ΔTinp, see TIMEAS) at that precise moment.

3. If the cell is in the CVLC charge mode, the interval that regulates the data-taking rate is not updated.

4. If the cell is in operation, the calibration factor of the controller is not updated.

D.7.30 MSWHEN

MSWHEN is called at the first pass of the stop or hold operation. It sends information to the WATCHDOG program, about how the cell is running. It also sends JCYCLE and KCYCLE, which are described in MSHOLD.

D.7.31 MSSTOP

MSSTOP is called at the second pass of the STOP operation It initiates a timer request for the cell.

D.7.32 CRSTOP

CRSTOP is a completion routine which is called when a timer request initiated by MSSTOP expires. IMSTOP is set equal to the number of the cell, for which the request was pending.

D.7.33 MSHOLD

MSHOLD is divided in two sections. The first section is called at the second pass of the HOLD operation. It determines if the HOLD request is valid, i.e., if the user does not ask to stop a cell at the end of a half-cycle, when it is already operating at a further half-cycle. Then it sets JCYCLE(J) and KCYCLE(J). When the cell is originally put in operation, KCYCLE(J) = -1 : that means that there is no HOLD request pending. MSHOLD sets JCYCLE(J) equal to the cycle number, and KCYCLE(J) to the mode of the last half-cycle before termination.

The second section is called when any cell goes in open-circuit. In that case, CHANGE has set IHOLD = 1, which was detected by MSBOSS. MSHOLD checks if any of these cells (among which the one that just went in open-circuit) has a HOLD request pending. In that case, it sets A(1,J)=2 for the cell (and for its eventual companion).

D.7.34 MSNOW

MSNOW is called when MSBOSS detects that the stop flag IMSTOP was set by CRSTOP. It sets A(1,J)=2 for the cell (and for its eventual companion).

D.7.35 HDCONC

HDCONC is a concatenated file of all the I/O library routines : SET, GET, PLACE, IMPOSE, RLOGG, EXT, RMLOGG. All these routines are described in detail in Appendix C. One additional I/O routine is provided, RDLOGG

D.7.36 RDLOGG (PULSED CURRENT CONTROL MODE)

RDLOGG is used for reading an A/D channel. It decides whether it should use RLOGG or RMLOGG. RLOGG is the 'regular' A/D reading routine. RMLOGG is specifically used when the cell operates in the pulsating current mode (see EXCON). The I/O driver follows the pulse and measures minimum (MAXMIN=0) or maximum (MAXMIN=1) values. It measures the cell voltage and (eventually) the reference voltages at the OFF-side of the pulse (MAXMIN=0), and the cell voltage at the ON-side of the pulse (MAXMIN=1). As explained in EXCON, this last measurement is automatically assigned to channel 5 (when parameters for the cell are initially entered). It measures on the same A/D channel as channel 1 (cell voltage). The A/D range used is also the same, but to differentiate between the 2 measurements, 100 is added to it (e.g., range 2 becomes 102). This is all decoded in RDLOGG.

The current is measured on a current averager channel. This is a DC measurement; therefore, it uses subroutine RLOGG, not RMLOGG.

D.8 LIST OF GLOBAL PARAMETERS

```
COMMON/CHACHA/AREAC
        AREAC(160):  Space allocated for 32 I/O channels (INTEGER).
COMMON/CHANNELS/ICHAN
        ICHAN(16) :  For each cell, the RT-11 channel through which
                     it communicates with its data-file (INTEGER).
COMMON/CCHOLD/JCYCLE,MCYCLE,IHOLD
        JCYCLE(16):  For each cell, on a HOLD request, the number
                     of the last cycle before termination. Not
                     defined if no request (INTEGER).
        KCYCLE(16):  For each cell, on a HOLD request, the mode
                     (DS:0, CH:1) of the last half-cycle before
                     termination. Is -1 if no request is pending
                     (INTEGER).
        IHOLD     :  Flag set (in CHANGE) when any cell goes in
                     open-circuit (INTEGER).
COMMON/CURCUR/CUR,COMP
        CUR(16)   :  Used in the current control routines (CC,
                     CVLC, ..) : for each cell, the last update of
                     the setpoint (in V) for its current controller
                     (REAL).
        COMP(16)  :  Used in the TIMEAS routine : for each cell, the
                     last measurement on the channel that is followed
                     for the adjustment of the rate (REAL).
COMMON/CURTIM/JLAST,JREM
        JLAST(16) :  Used in the current control routines (CC,...)
                     for each cell, the last time at which a current
                     regulation was performed (INTEGER*4).
        JREM(16)  :  Used in the TIMEAS routine : for each cell,
                     temporary storage of JT(4,J) (INTEGER*4).
COMMON/CYCLCT/ICYCLE,MCYCLE
        ICYCLE(16):  For each cell, the cycle at which it is running
                     (INTEGER).
        MCYCLE(16):  For each cell, 1 if a cycle starts with CH, 0 if
                     DS (INTEGER).
COMMON/GDNIT/JMIN
        JMIN      :  Minimum of all action timers (INTEGER*4).
COMMON/HNDLER/MCHAN
        MCHAN     :  Channel through which the program communicates
                     with the I/O driver HD (INTEGER).
COMMON/INT4/JTWO,JFOUR,JFAC
        JTWO      :  2 (INTEGER*4).
        JFOUR     :  4 (INTEGER*4).
        JFAC(16)  :  For each cell, the factor that cuts the data
                     storage time (INTEGER*4).
COMMON/MESSAG/MSG,MESCNT,AREA,AREA2,IMSTOP
        MSG(59)   :  Space allocated for incoming messages (INTEGER).
        MESCNT    :  Flag for message in (INTEGER).
        AREA(4)   :  Area for linkage information, used by the
                     completion routine call for CRMESS (INTEGER).
        AREA2(4)  :  idem for CRSTOP (INTEGER).
        IMSTOP    :  Flag set by CRSTOP, equal to the number of the
                     cell, which timer request expired (INTEGER).
```

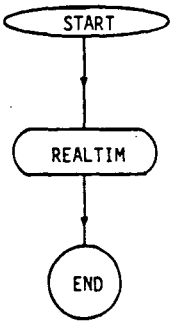
```
COMMON/OFFSET/OFF
        OFF(16)    :  For each cell, the offset of its current
                      controller (REAL).
COMMON/PASS/A,B,JT,VAL,Q,QO
                   : Parameter arrays, described in Section D.1.
COMMON/PNTRS/KPOINT,LPOINT
        KPOINT     : Pointer of FMAT.UNF (INTEGER).
        LPOINT     : Pointer of GMAT.UNF (INTEGER).
COMMON/QUEUE/AREAQ
        AREAQ(100): Space allocated for 10 queue elements (INTEGER).
COMMON/RES/JTOP,JMAX
        JTOP       : Time of day (INTEGER*4).
        JMAX       : Maximum number of cells, default is 16 (INTEGER).
COMMON/TIMTIM/JCHECK,JSET,JCVLC
        JCHECK     : 1 minute   (INTEGER*4).
        JSET       : 1 minute   (INTEGER*4).
        JCVLC      : 15 seconds (INTEGER*4).

BUF(128,16)        : Storage area for each cell
```

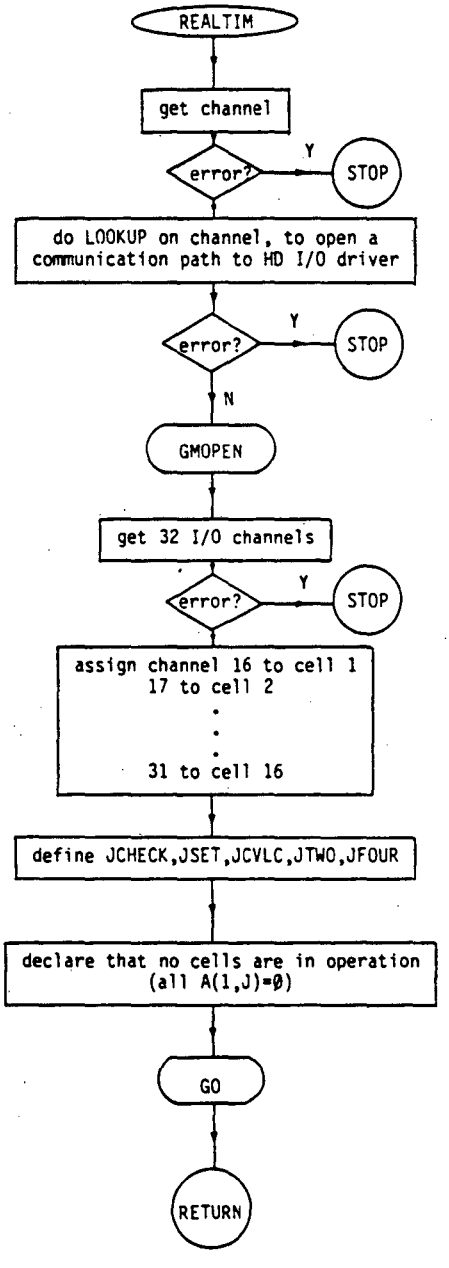
D.9 STARTING THE REAL-TIME PROGRAM; EMERGENCY STOP

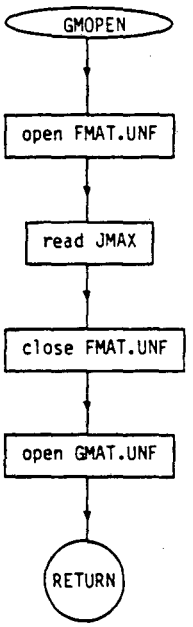
To start the REAL-TIME program, the user must type @DLØ:REALTS on the BG terminal. The REAL-TIME program and the WATCHDOG program start running. An emergency stop for the REAL-TIME program is a ^C^C typed on the FG-terminal. It is safer, however, if there is enough time, to stop all cells separately (STOP request see Appendix E).

D.10 CODE

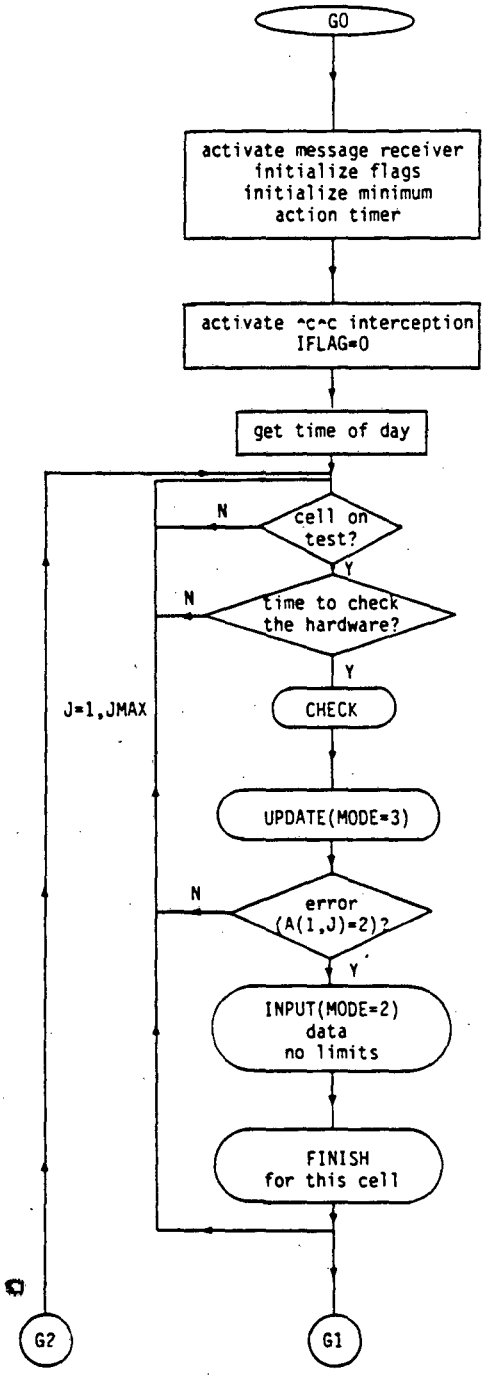
START
REALTIM



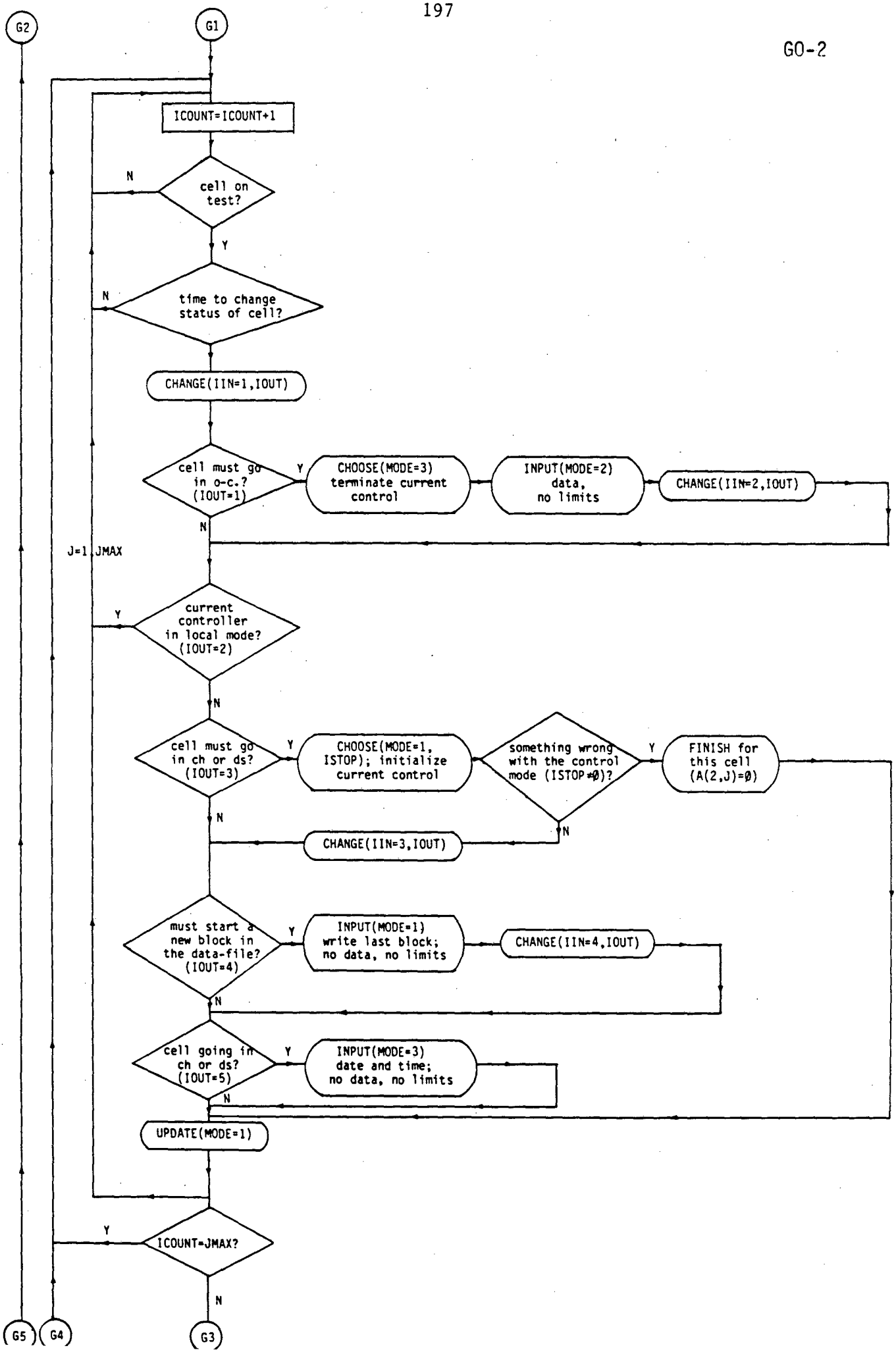


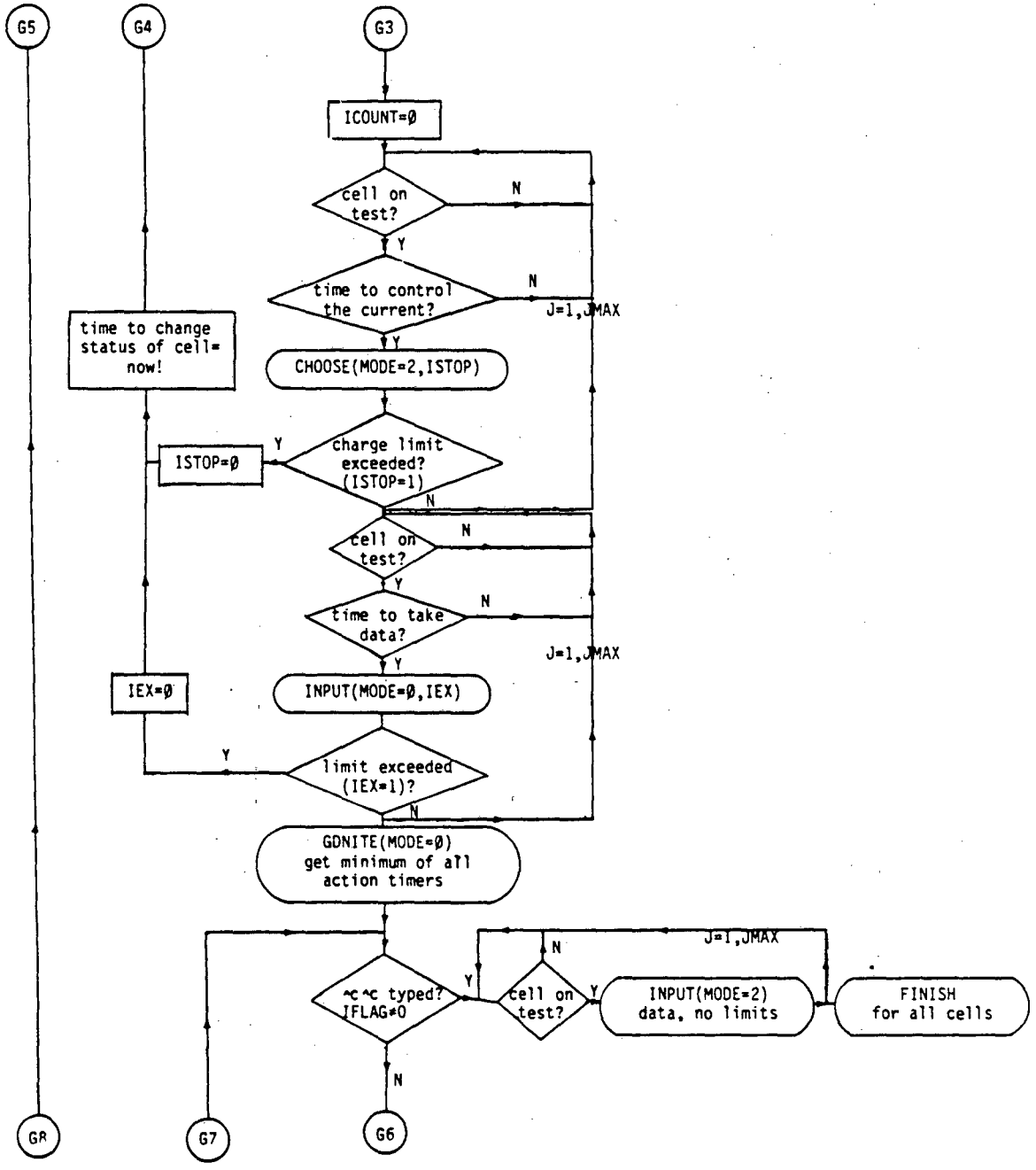

GMOPEN
open FMAT.UNF
read JMAX
close FMAT.UNF
open GMAT.UNF
RETURN

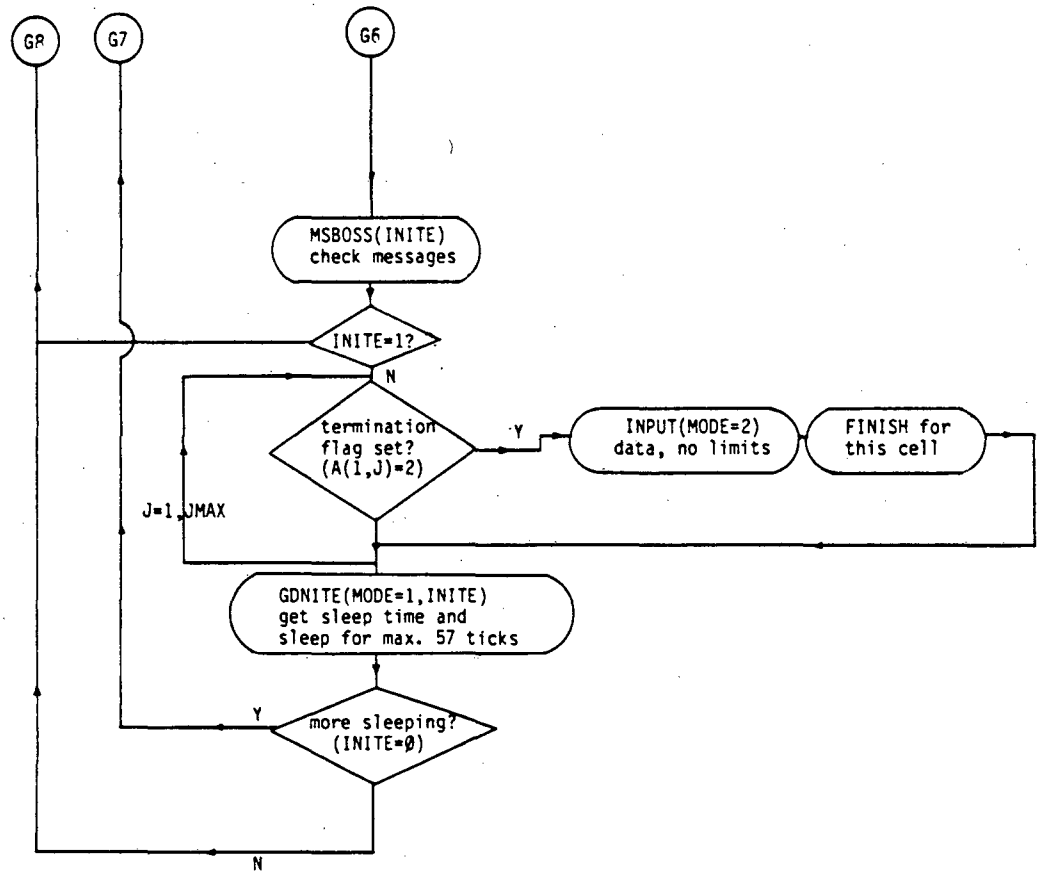
GO-1


GO
activate message receiver
initialize flags
initialize minimum
action timer
activate ^c^c interception
IFLAG=0
get time of day
N
cell on
test?
Y
N
time to check
the hardware?
Y
J=1,JMAX
CHECK
UPDATE(MODE=3)
N
error
(A(1,J)=2)?
Y
INPUT(MODE=2)
data
no limits
FINISH
for this cell
G2
G1

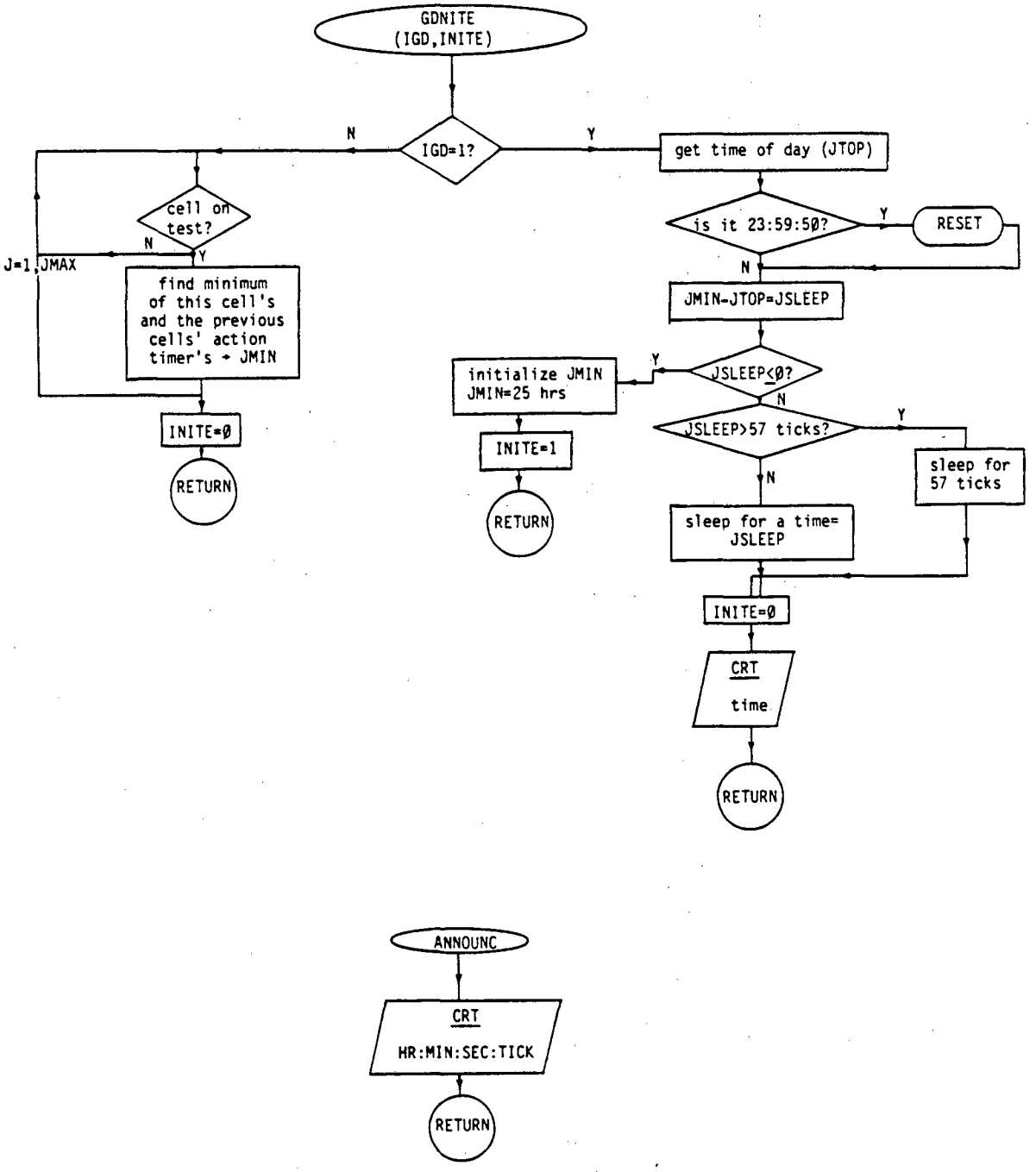
GO-2

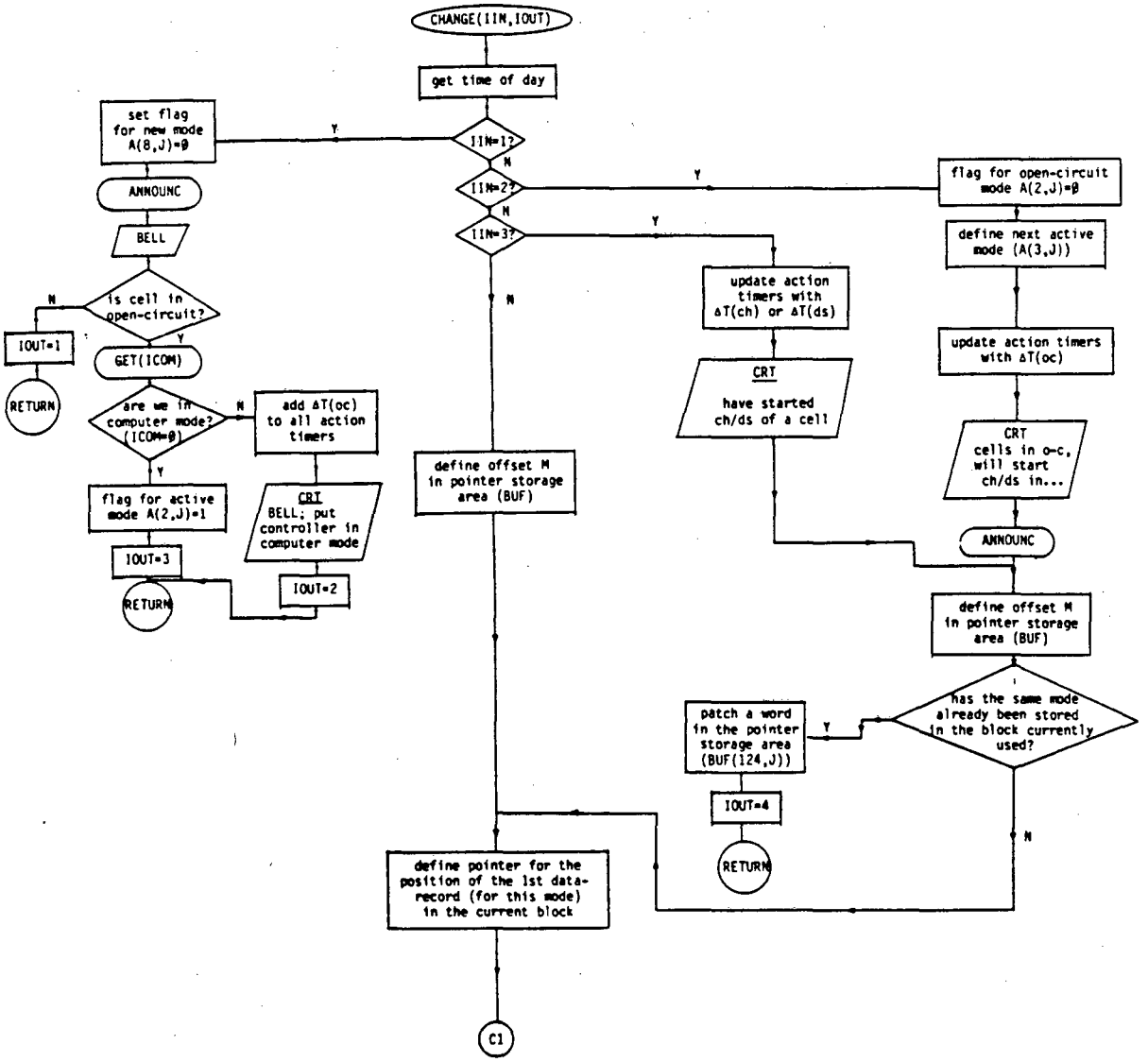

G2
G1
ICOUNT=ICOUNT+1
N
cell on test?
Y
N
time to change status of cell?
CHANGE(IIN=1,IOUT)
cell must go in o-c.? (IOUT=1)
Y
CHOOSE(MODE=3) terminate current control
INPUT(MODE=2) data, no limits
CHANGE(IIN=2,IOUT)
N
J=1,JMAX
Y
current controller in local mode? (IOUT=2)
N
cell must go in ch or ds? (IOUT=3)
Y
CHOOSE(MODE=1, ISTOP); initialize current control
something wrong with the control mode (ISTOP≠Ø)?
Y
FINISH for this cell (A(2,J)=Ø)
N
N
CHANGE(IIN=3,IOUT)
must start a new block in the data-file? (IOUT=4)
Y
INPUT(MODE=1) write last block; no data, no limits
CHANGE(IIN=4,IOUT)
N
cell going in ch or ds? (IOUT=5)
Y
INPUT(MODE=3) date and time; no data, no limits
N
UPDATE(MODE=1)
Y
ICOUNT=JMAX?
N
G5
G4
G3


G5
G4
G3
ICOUNT=Ø
cell on test?
N
Y
time to control the current?
N
J=1,JMAX
Y
time to change status of cell= now!
CHOOSE(MODE=2,ISTOP)
ISTOP=Ø
Y
charge limit exceeded? (ISTOP=1)
N
cell on test?
N
Y
time to take data?
N
J=1,JMAX
Y
IEX=Ø
INPUT(MODE=Ø,IEX)
Y
limit exceeded (IEX=1)?
N
GDNITE(MODE=Ø) get minimum of all action timers
J=1,JMAX
N
^C^C typed? IFLAG≠0
Y
cell on test?
Y
INPUT(MODE=2) data, no limits
FINISH for all cells
N
GR
G7
G6

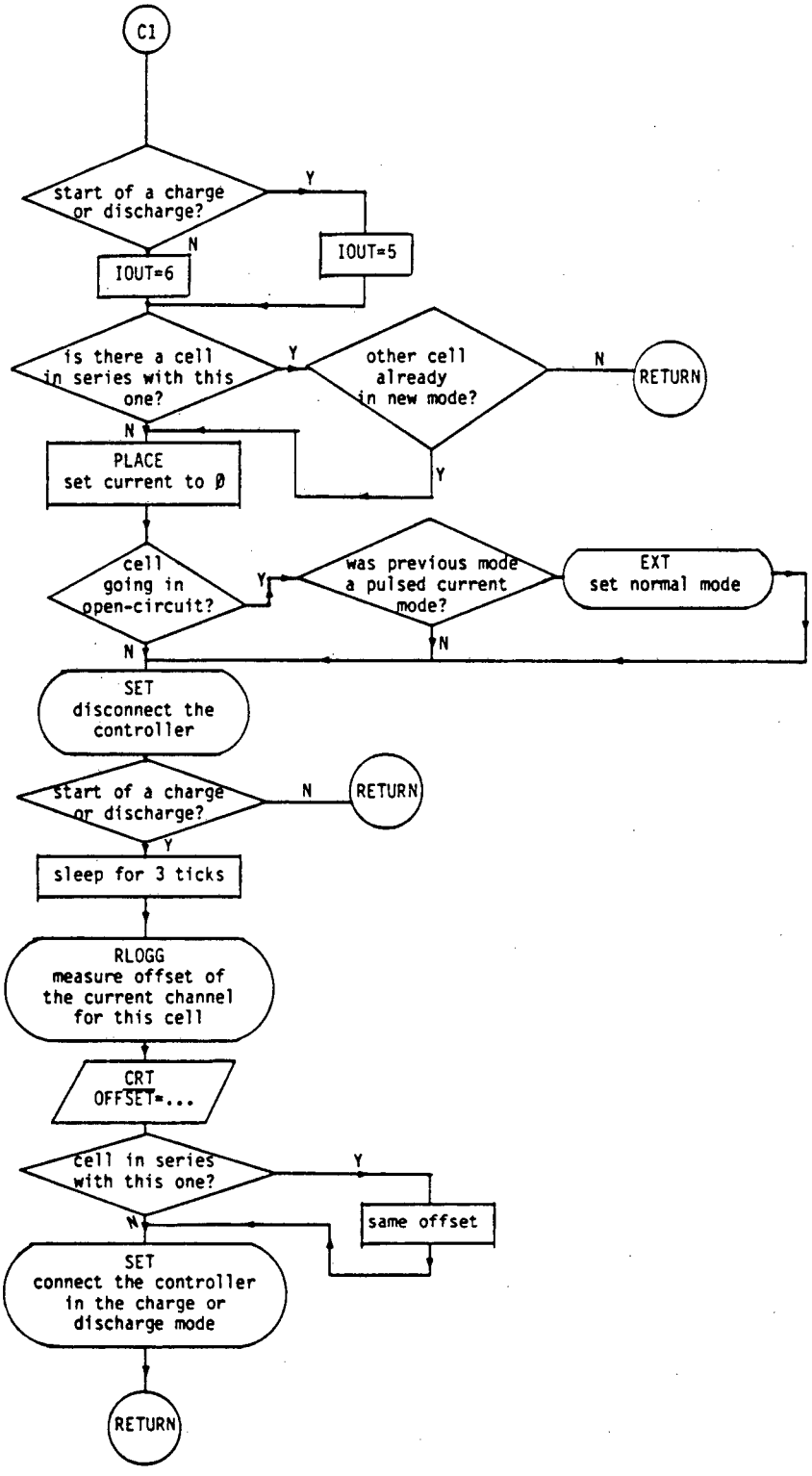

GR
G7
G6
MSBOSS(INITE)
check messages
INITE=1?
N
termination
flag set?
(A(I,J)=2)
Y
INPUT(MODE=2)
data, no limits
FINISH for
this cell
J=1,JMAX
GDNITE(MODE=1,INITE)
get sleep time and
sleep for max. 57 ticks
Y
more sleeping?
(INITE=Ø)
N

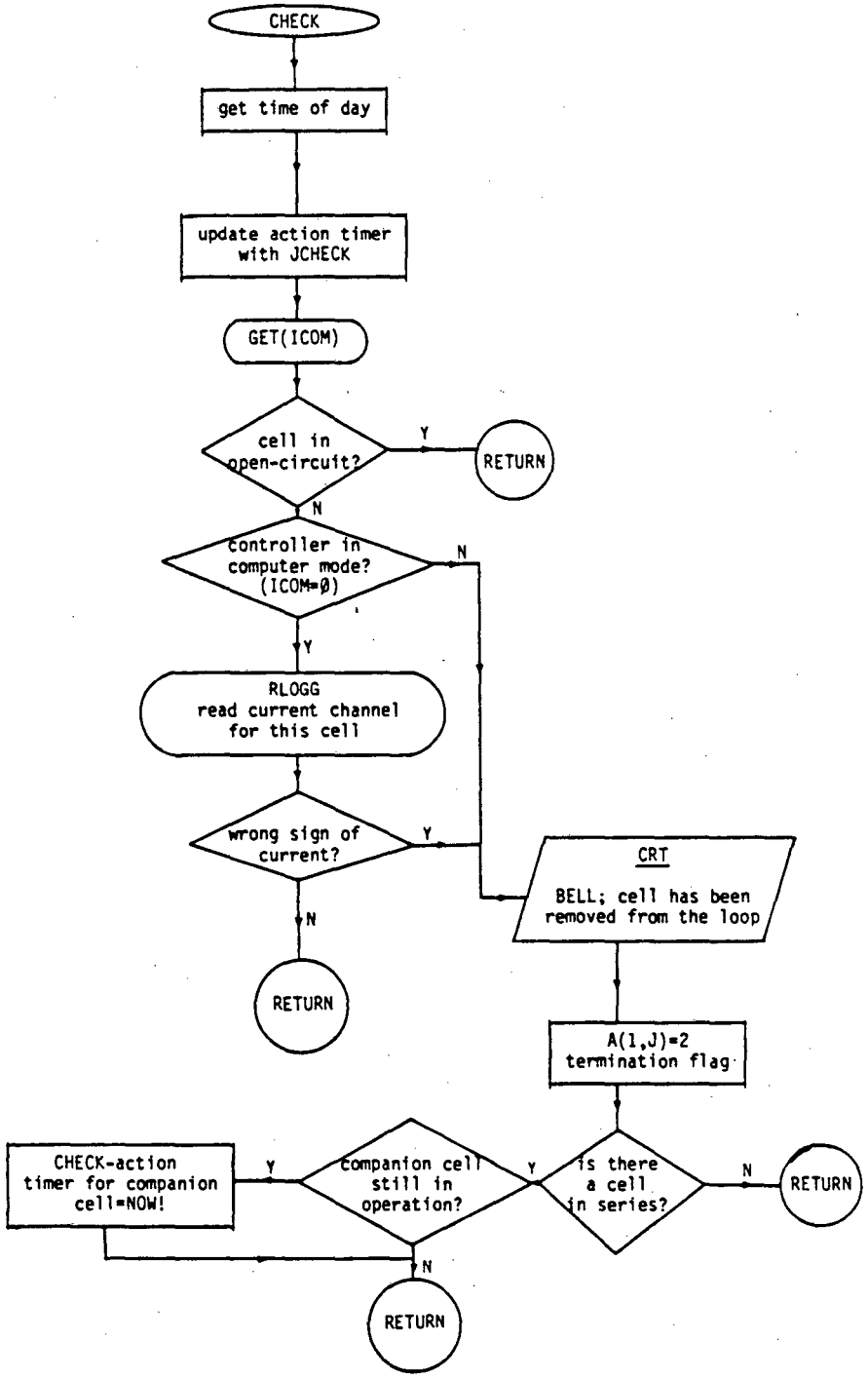
GDNITE
ANNOUNC

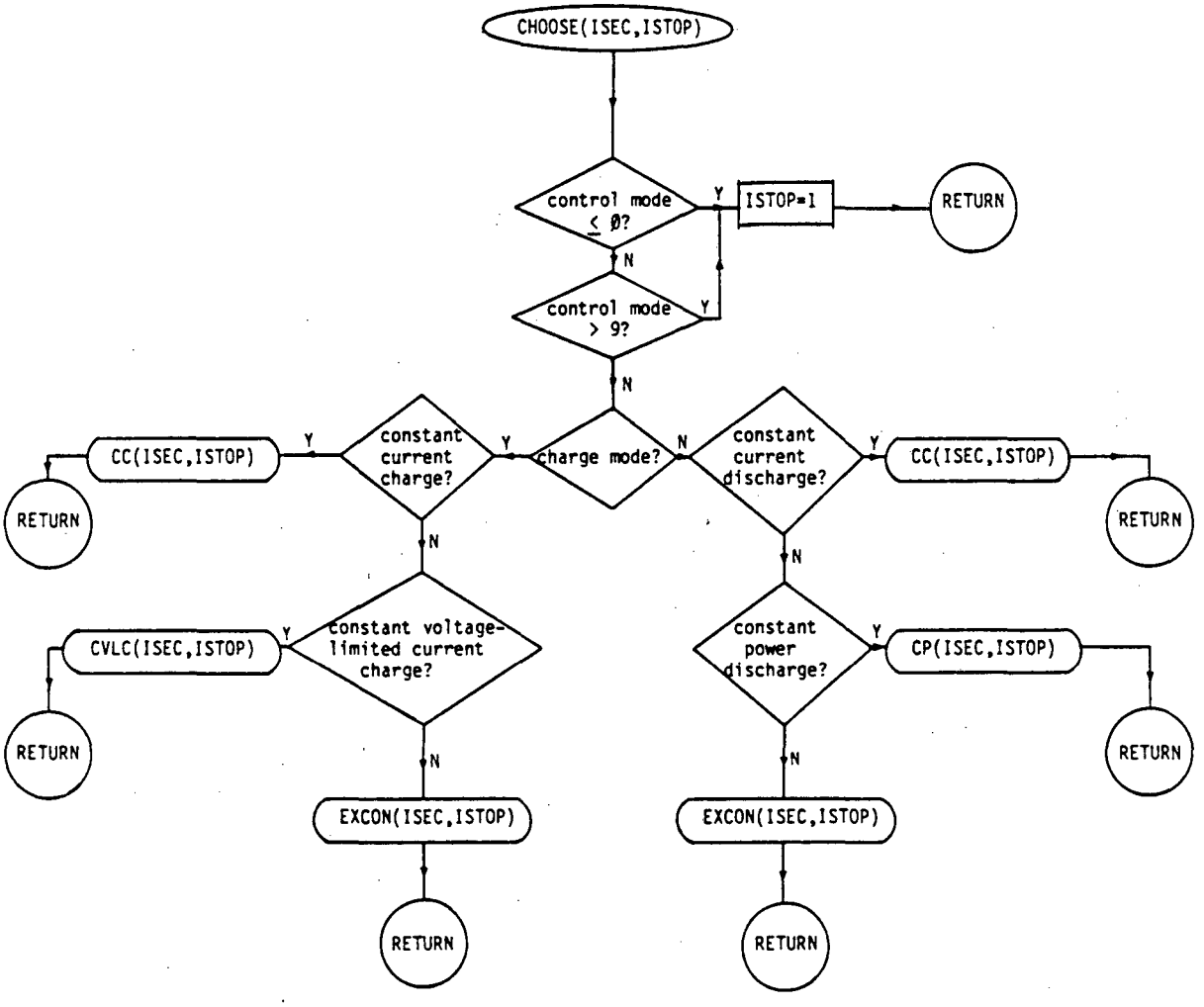



CHANGE(IIN,IOUT)
get time of day
IIN=1?
Y
set flag for new mode A(8,J)=0
ANNOUNC
BELL
N
is cell in open-circuit?
Y
IOUT=1
RETURN
GET(ICOM)
are we in computer mode? (ICOM=0)
N
add ΔT(oc) to all action timers
Y
flag for active mode A(2,J)=1
CRT
BELL; put controller in computer mode
IOUT=3
IOUT=2
RETURN
N
IIN=2?
Y
flag for open-circuit mode A(2,J)=0
define next active mode (A(3,J))
update action timers with ΔT(oc)
CRT
cells in o-c, will start ch/ds in...
ANNOUNC
define offset M in pointer storage area (BUF)
has the same mode already been stored in the block currently used?
Y
patch a word in the pointer storage area (BUF(124,J))
IOUT=4
RETURN
N
IIN=3?
Y
update action timers with ΔT(ch) or ΔT(ds)
CRT
have started ch/ds of a cell
N
define offset M in pointer storage area (BUF)
define pointer for the position of the 1st data-record (for this mode) in the current block
C1

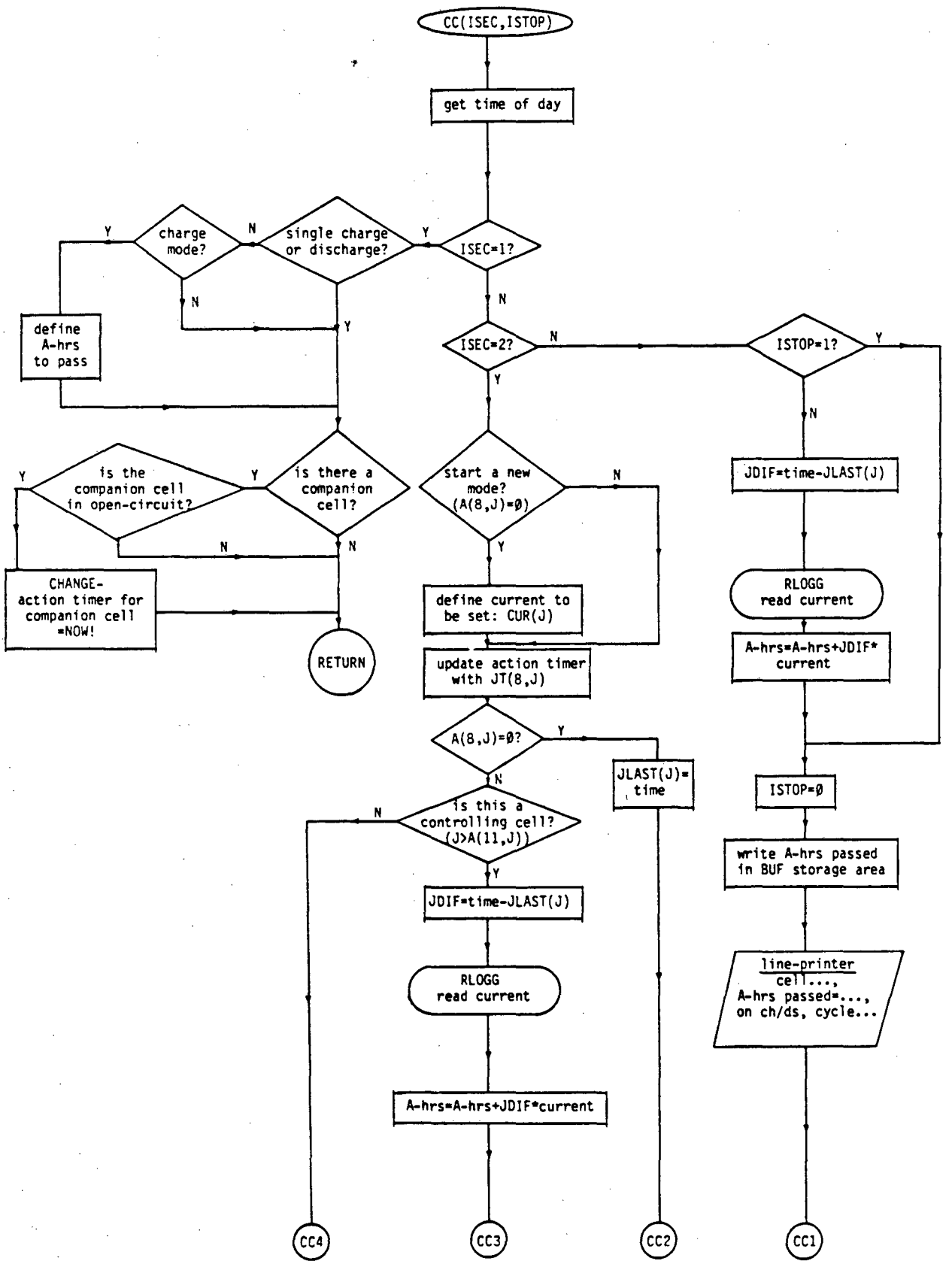
CHANGE-2


C1
start of a charge or discharge?
Y
N
IOUT=6
IOUT=5
is there a cell in series with this one?
Y
other cell already in new mode?
N
RETURN
N
Y
PLACE set current to Ø
cell going in open-circuit?
Y
was previous mode a pulsed current mode?
EXT set normal mode
N
N
SET disconnect the controller
start of a charge or discharge?
N
RETURN
Y
sleep for 3 ticks
RLOGG measure offset of the current channel for this cell
CRT OFFSET=...
cell in series with this one?
Y
same offset
N
SET connect the controller in the charge or discharge mode
RETURN

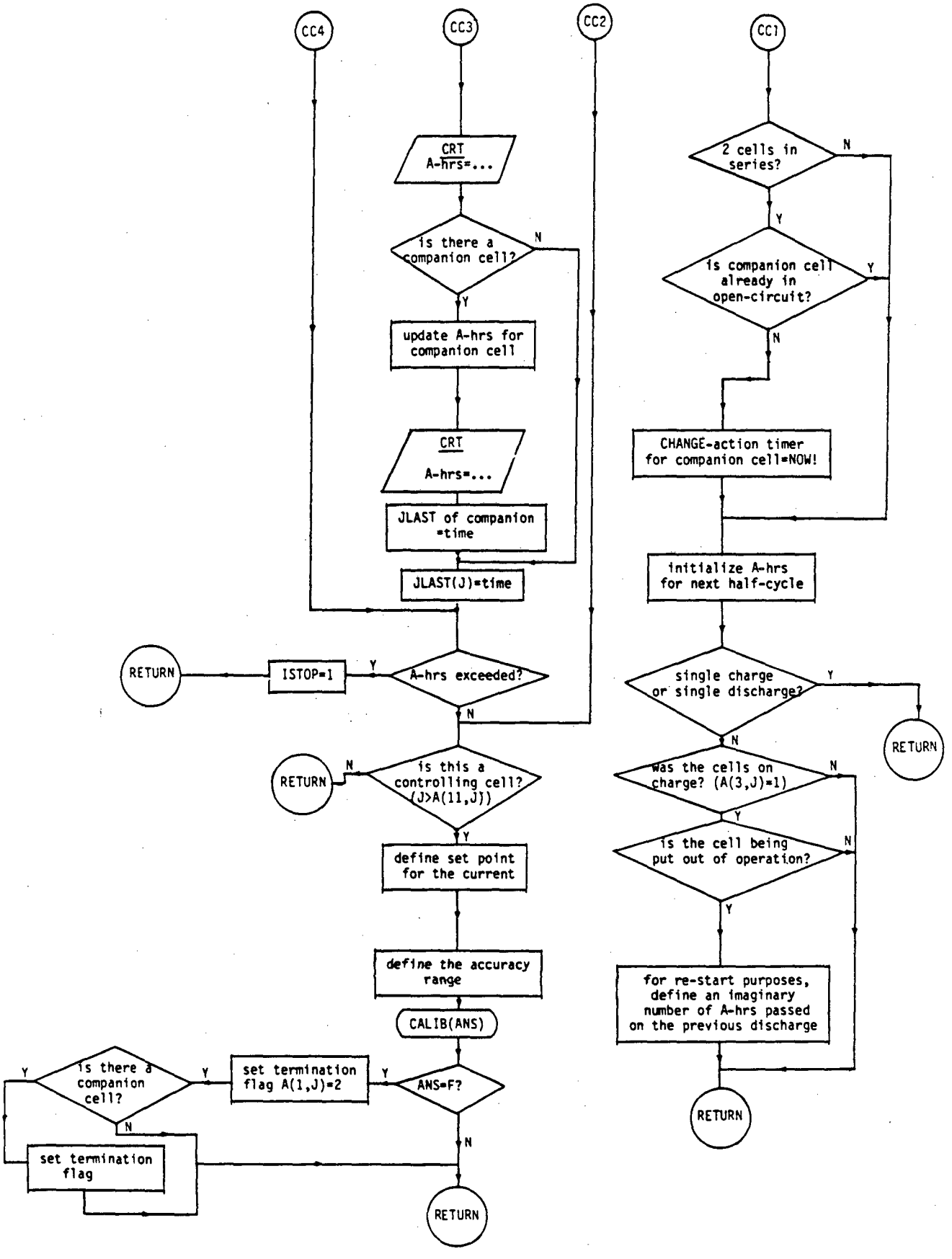

CHECK
get time of day
update action timer with JCHECK
GET(ICOM)
cell in open-circuit?
Y
RETURN
N
controller in computer mode? (ICOM=Ø)
N
Y
RLOGG
read current channel for this cell
wrong sign of current?
Y
N
RETURN
CRT
BELL; cell has been removed from the loop
A(1,J)=2
termination flag
is there a cell in series?
N
RETURN
Y
companion cell still in operation?
Y
CHECK-action timer for companion cell=NOW!
N
RETURN


CHOOSE(ISEC,ISTOP)
control mode ≤ 0?
Y
ISTOP=1
RETURN
N
control mode > 9?
Y
N
charge mode?
constant current charge?
Y
CC(ISEC,ISTOP)
RETURN
N
constant voltage-limited current charge?
CVLC(ISEC,ISTOP)
RETURN
N
EXCON(ISEC,ISTOP)
RETURN
constant current discharge?
CC(ISEC,ISTOP)
RETURN
N
constant power discharge?
CP(ISEC,ISTOP)
RETURN
N
EXCON(ISEC,ISTOP)
RETURN

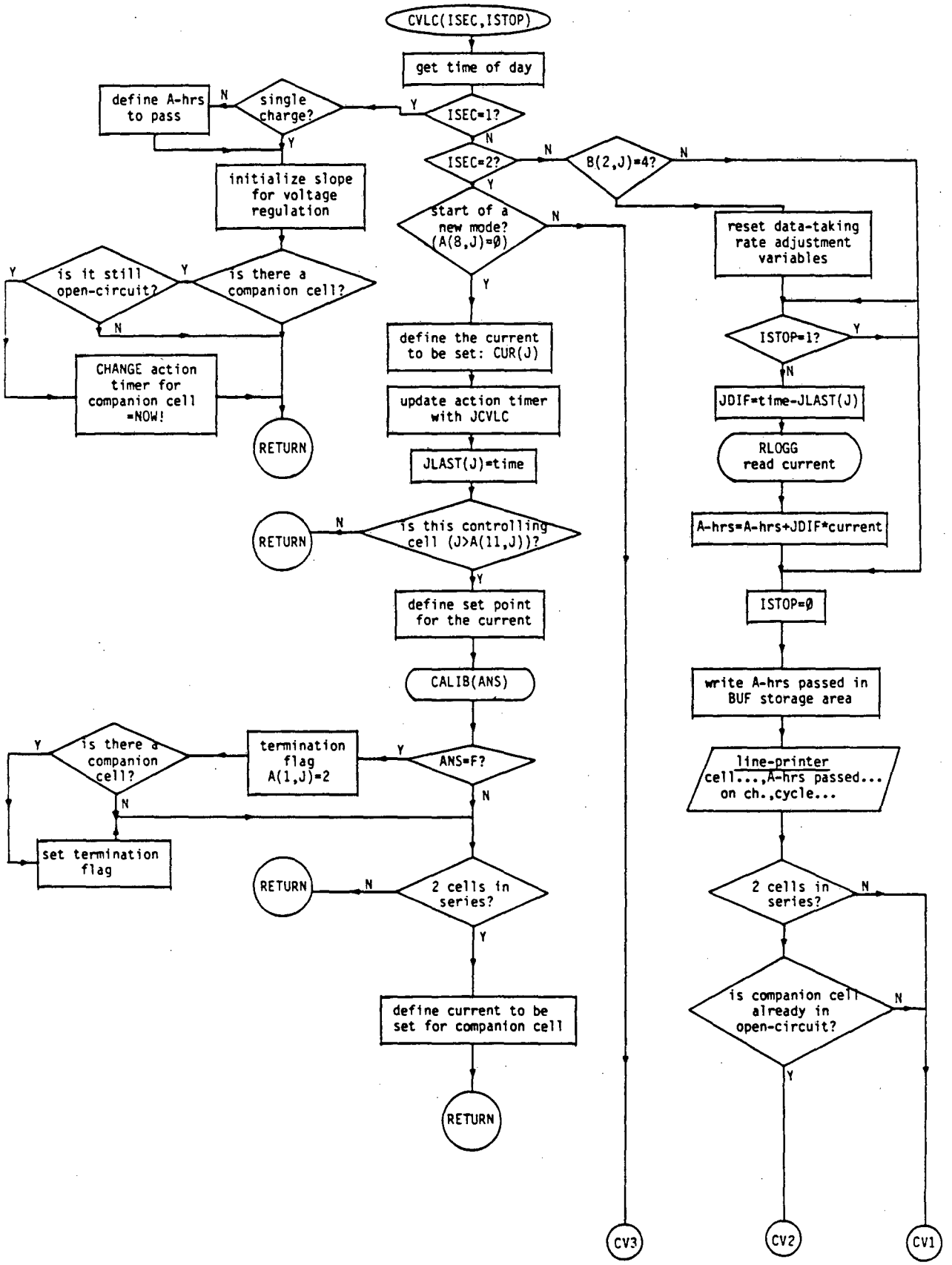
CC-1

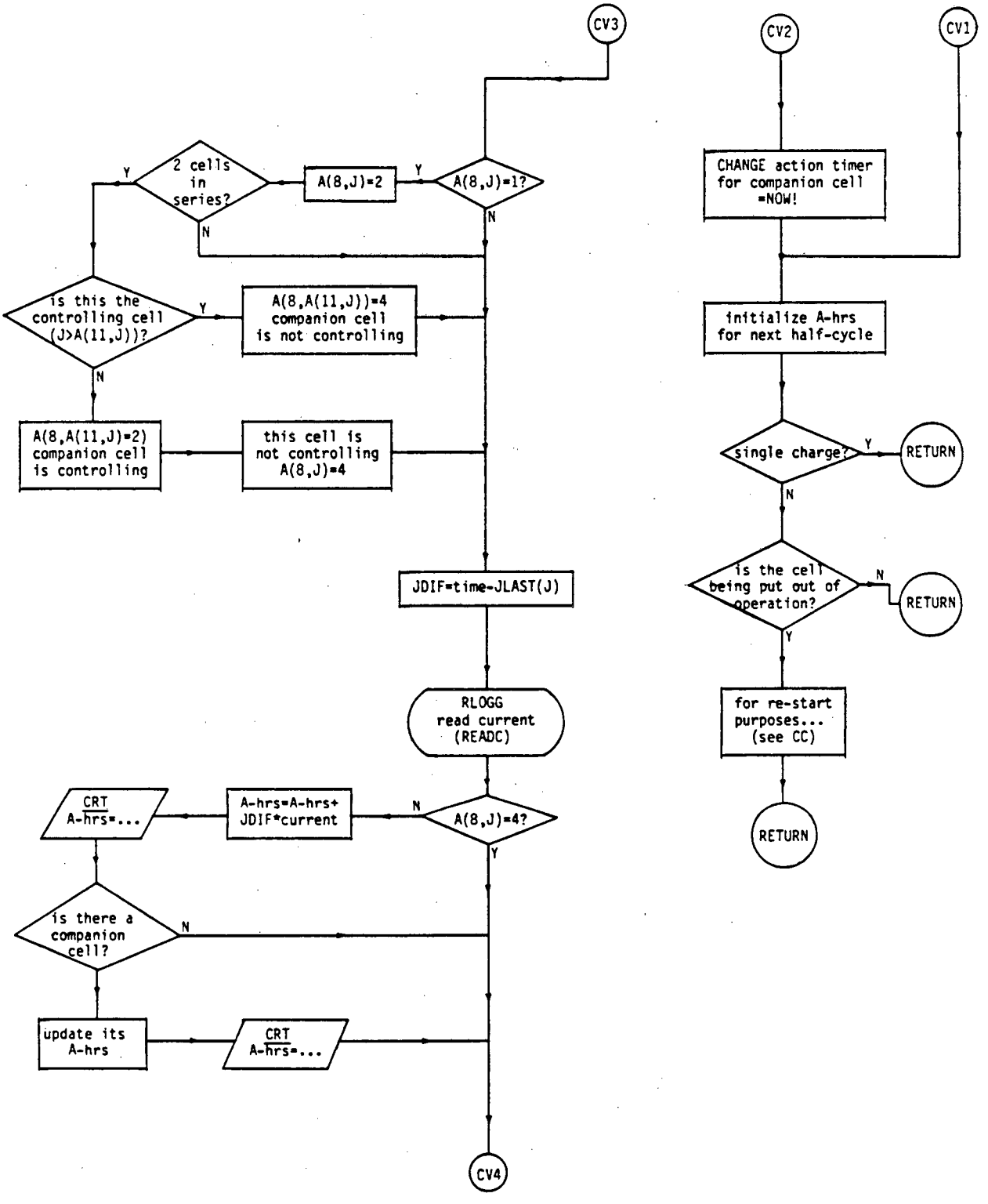

CC(ISEC,ISTOP)
get time of day
ISEC=1?
single charge or discharge?
charge mode?
define A-hrs to pass
ISEC=2?
ISTOP=1?
is there a companion cell?
is the companion cell in open-circuit?
CHANGE- action timer for companion cell =NOW!
RETURN
start a new mode? (A(8,J)=Ø)
define current to be set: CUR(J)
update action timer with JT(8,J)
A(8,J)=Ø?
JLAST(J)= time
is this a controlling cell? (J>A(11,J))
JDIF=time-JLAST(J)
RLOGG read current
A-hrs=A-hrs+JDIF*current
JDIF=time-JLAST(J)
RLOGG read current
A-hrs=A-hrs+JDIF* current
ISTOP=Ø
write A-hrs passed in BUF storage area
line-printer cell..., A-hrs passed=..., on ch/ds, cycle...
CC4
CC3
CC2
CC1

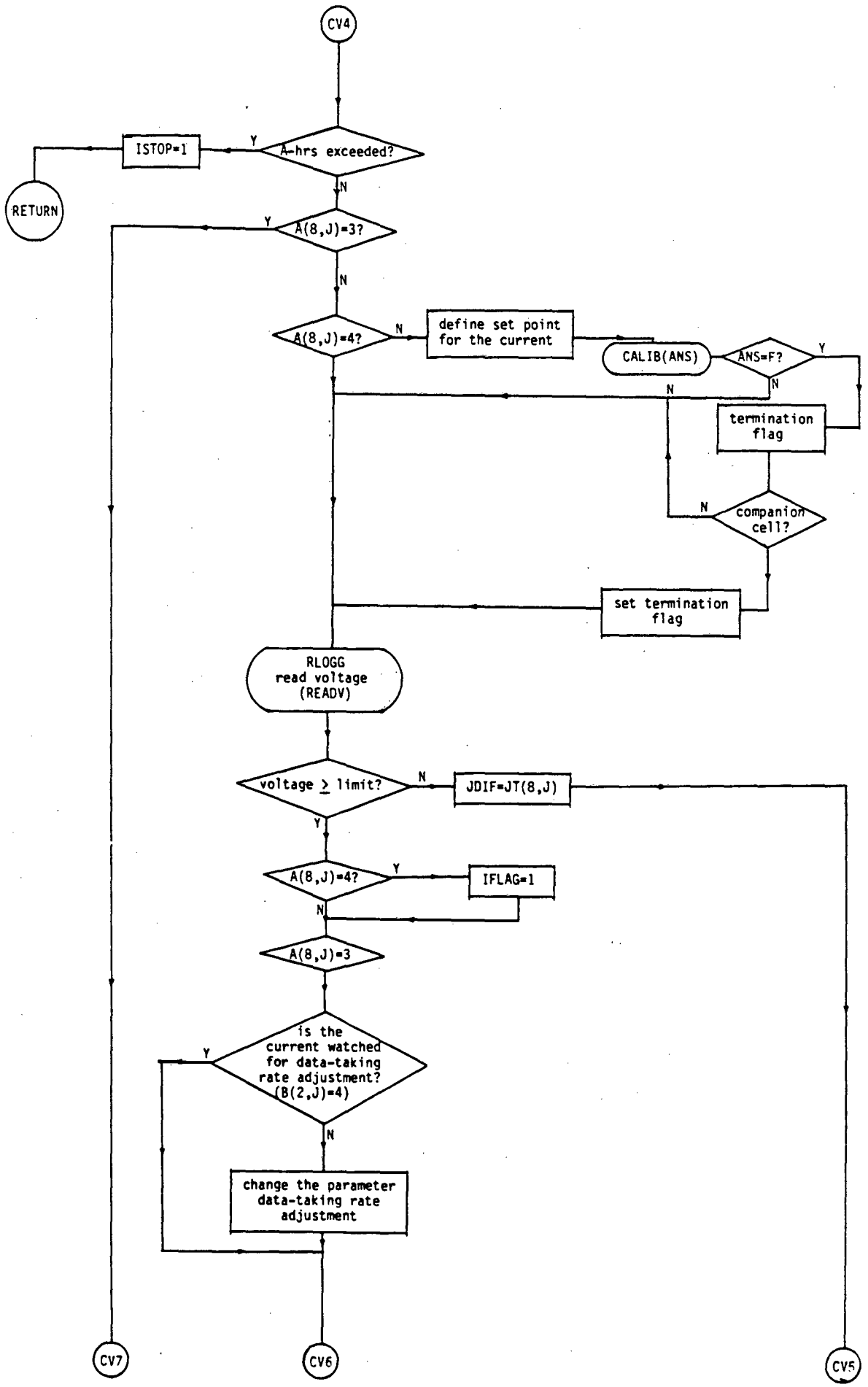
CC-2

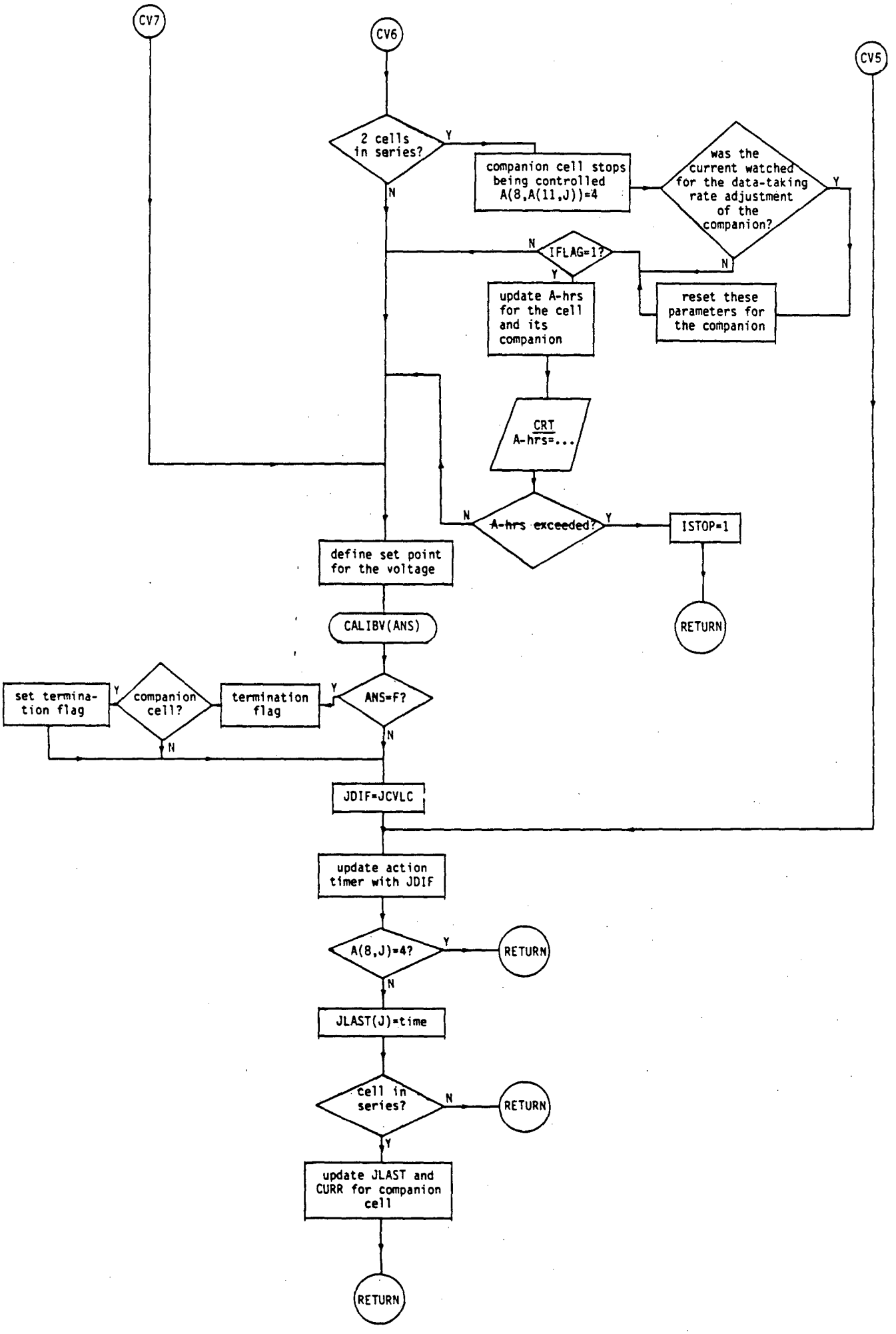

CC4
CC3
CC2
CC1
CRT
A-hrs=...
is there a companion cell?
N
Y
update A-hrs for companion cell
CRT
A-hrs=...
JLAST of companion =time
JLAST(J)=time
RETURN
ISTOP=1
Y
A-hrs exceeded?
N
RETURN
N
is this a controlling cell? (J>A(11,J))
Y
define set point for the current
define the accuracy range
CALIB(ANS)
Y
is there a companion cell?
Y
set termination flag A(1,J)=2
Y
ANS=F?
N
N
set termination flag
RETURN
2 cells in series?
N
Y
is companion cell already in open-circuit?
Y
N
CHANGE-action timer for companion cell=NOW!
initialize A-hrs for next half-cycle
single charge or single discharge?
Y
RETURN
N
was the cells on charge? (A(3,J)=1)
N
Y
is the cell being put out of operation?
N
Y
for re-start purposes, define an imaginary number of A-hrs passed on the previous discharge
RETURN

CVLC-1

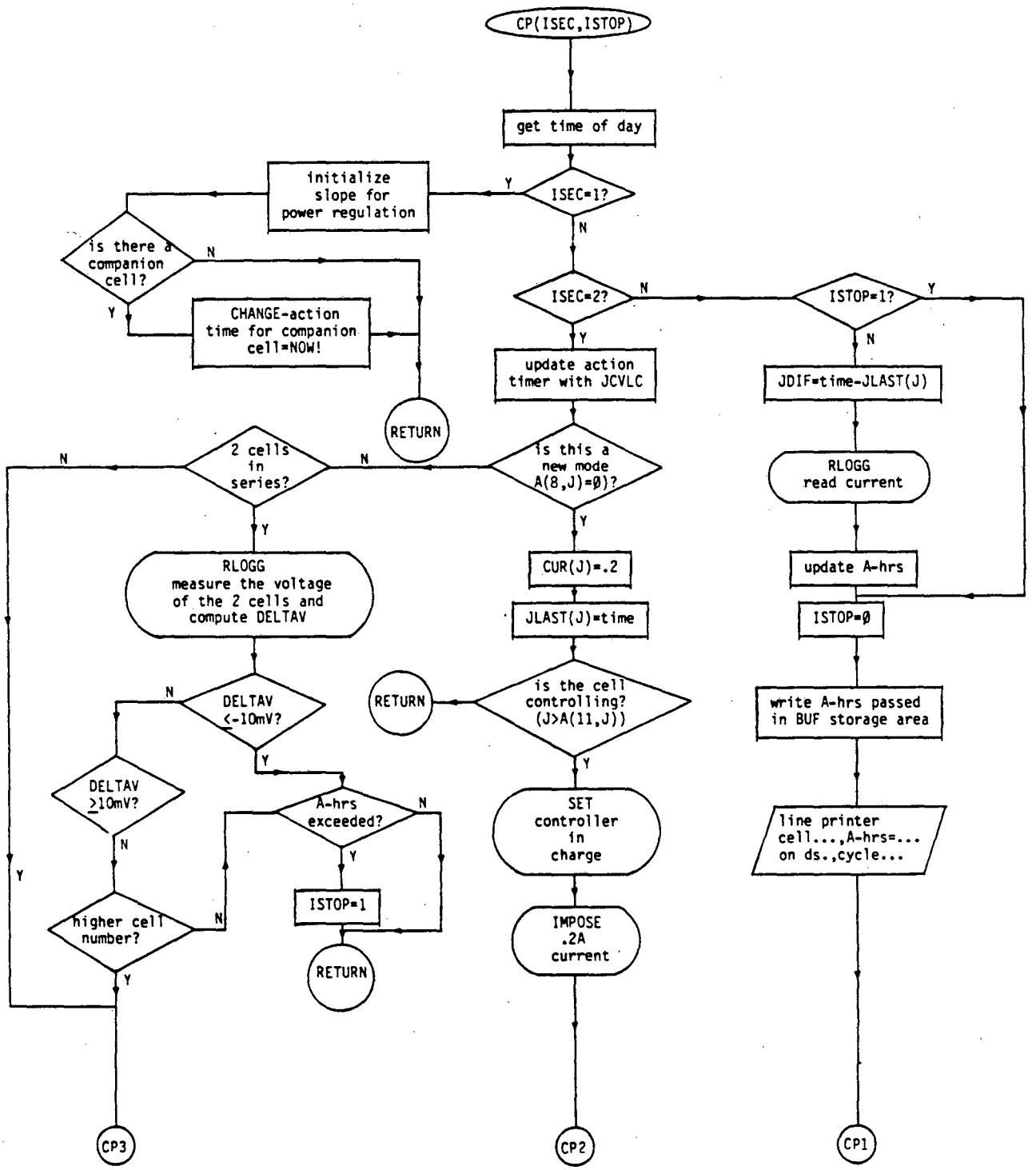

CVLC(ISEC,ISTOP)
get time of day
ISEC=1?
Y
N
single charge?
N
define A-hrs to pass
Y
initialize slope for voltage regulation
ISEC=2?
N
B(2,J)=4?
N
Y
start of a new mode? (A(8,J)=Ø)
N
Y
reset data-taking rate adjustment variables
is it still open-circuit?
Y
is there a companion cell?
Y
N
CHANGE action timer for companion cell =NOW!
RETURN
define the current to be set: CUR(J)
update action timer with JCVLC
JLAST(J)=time
RETURN
N
is this controlling cell (J>A(11,J))?
Y
define set point for the current
CALIB(ANS)
ISTOP=1?
Y
N
JDIF=time-JLAST(J)
RLOGG read current
A-hrs=A-hrs+JDIF*current
ISTOP=Ø
write A-hrs passed in BUF storage area
line-printer cell...,A-hrs passed... on ch.,cycle...
is there a companion cell?
Y
N
termination flag A(1,J)=2
ANS=F?
Y
N
set termination flag
RETURN
N
2 cells in series?
Y
define current to be set for companion cell
RETURN
2 cells in series?
N
is companion cell already in open-circuit?
N
Y
CV3
CV2
CV1


CV3
2 cells in series?
Y
N
A(8,J)=2
A(8,J)=1?
Y
N
is this the controlling cell (J>A(11,J))?
Y
N
A(8,A(11,J))=4 companion cell is not controlling
A(8,A(11,J)=2) companion cell is controlling
this cell is not controlling A(8,J)=4
JDIF=time-JLAST(J)
RLOGG read current (READC)
A(8,J)=4?
N
Y
A-hrs=A-hrs+ JDIF*current
CRT A-hrs=...
is there a companion cell?
N
update its A-hrs
CRT A-hrs=...
CV4
CV2
CV1
CHANGE action timer for companion cell =NOW!
initialize A-hrs for next half-cycle
single charge?
Y
RETURN
N
is the cell being put out of operation?
N
RETURN
Y
for re-start purposes... (see CC)
RETURN

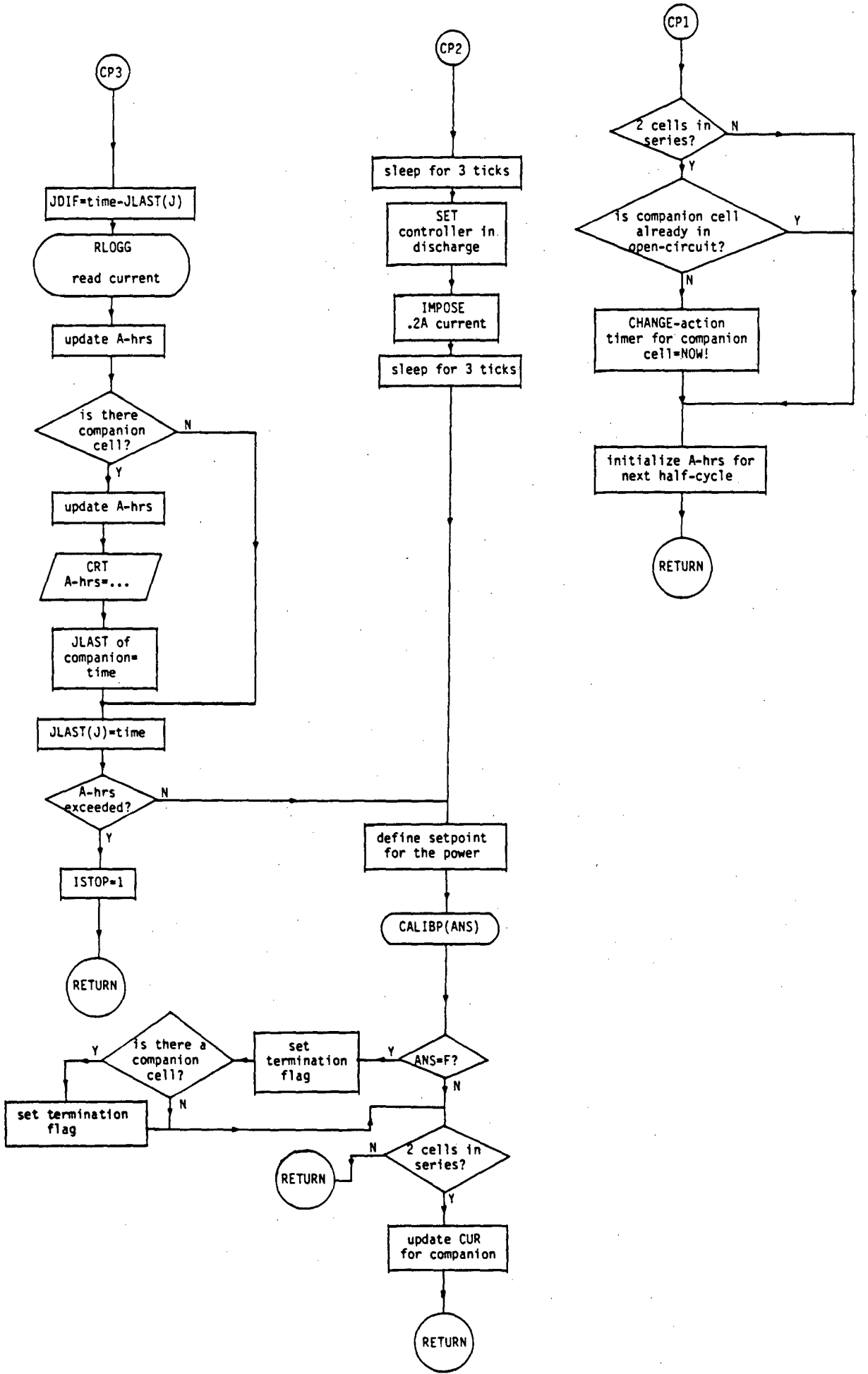

CV4
A-hrs exceeded?
Y
ISTOP=1
RETURN
N
A(8,J)=3?
Y
N
A(8,J)=4?
N
define set point for the current
CALIB(ANS)
ANS=F?
Y
N
N
termination flag
N
companion cell?
set termination flag
RLOGG read voltage (READV)
voltage ≥ limit?
N
JDIF=JT(8,J)
Y
A(8,J)=4?
Y
IFLAG=1
N
A(8,J)=3
is the current watched for data-taking rate adjustment? (B(2,J)=4)
Y
N
change the parameter data-taking rate adjustment
CV7
CV6
CV5


CV7
CV6
CV5
2 cells in series?
Y
N
companion cell stops being controlled A(8,A(11,J))=4
was the current watched for the data-taking rate adjustment of the companion?
Y
N
N
IFLAG=1?
Y
N
update A-hrs for the cell and its companion
reset these parameters for the companion
CRT
A-hrs=...
N
A-hrs exceeded?
Y
ISTOP=1
RETURN
define set point for the voltage
CALIBV(ANS)
set termination flag
Y
companion cell?
Y
N
termination flag
Y
ANS=F?
N
JDIF=JCVLC
update action timer with JDIF
A(8,J)=4?
Y
RETURN
N
JLAST(J)=time
cell in series?
N
RETURN
Y
update JLAST and CURR for companion cell
RETURN

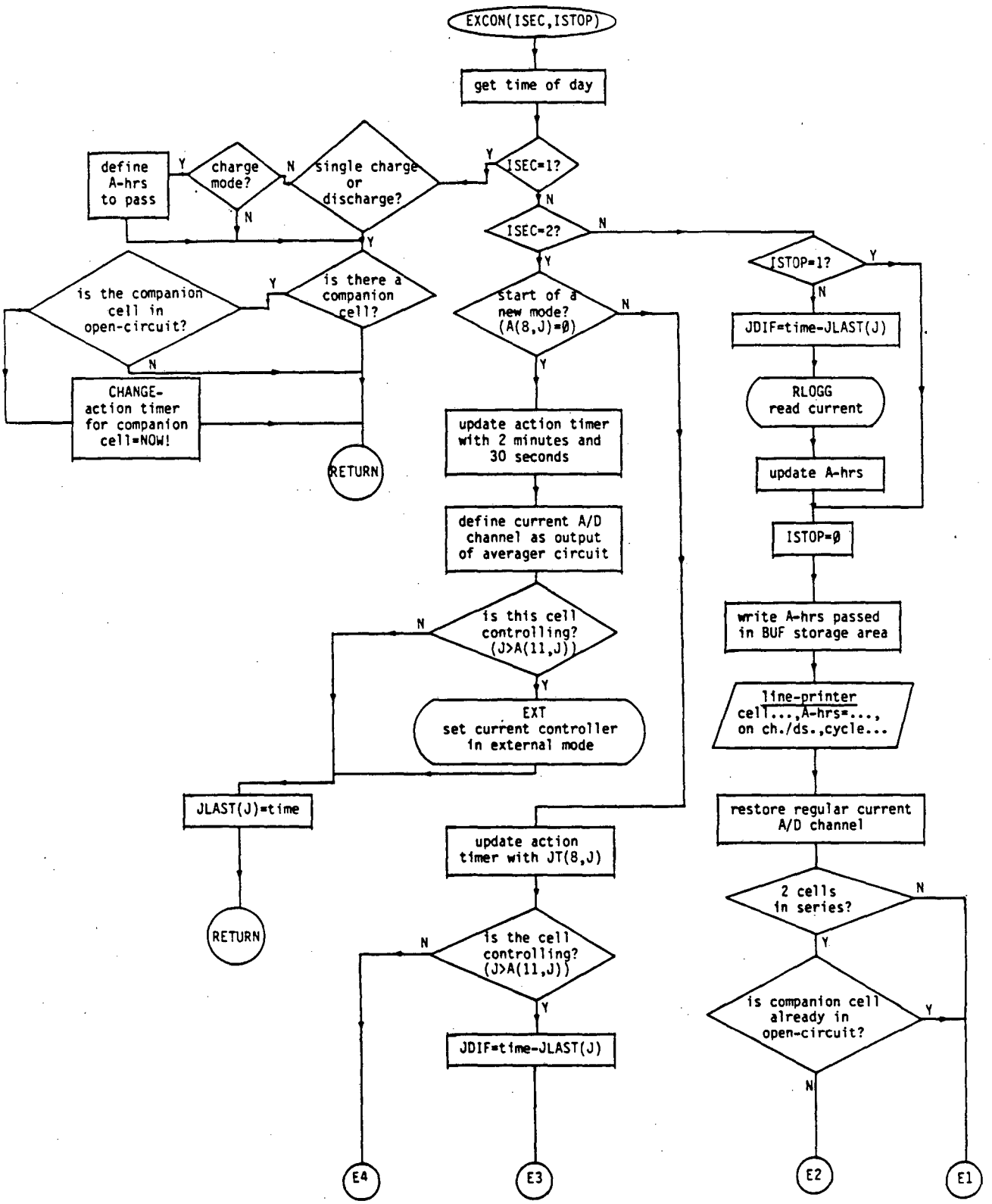
CP-1

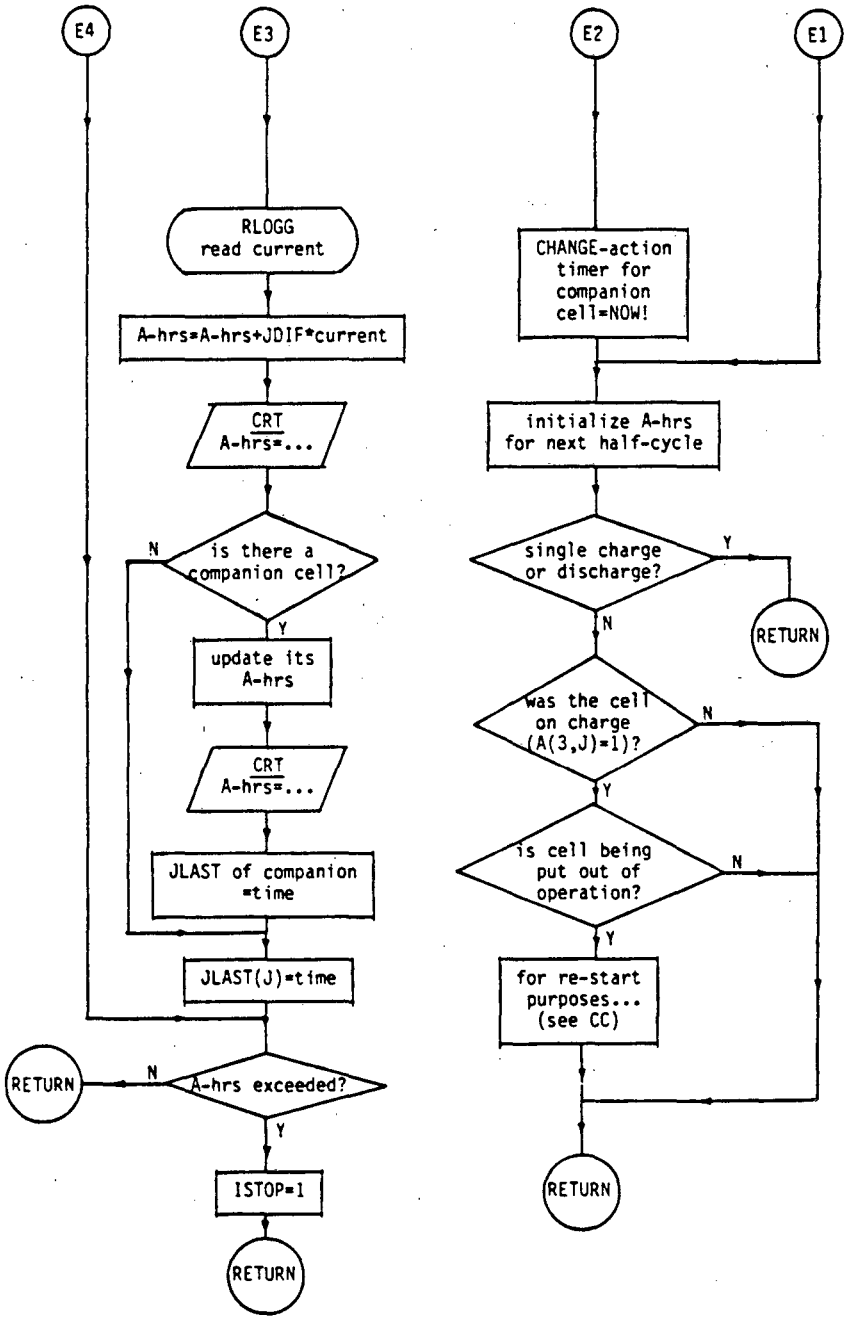

CP(ISEC,ISTOP)
get time of day
ISEC=1?
Y
initialize slope for power regulation
N
ISEC=2?
N
ISTOP=1?
Y
N
is there a companion cell?
N
Y
CHANGE-action time for companion cell=NOW!
RETURN
Y
update action timer with JCVLC
JDIF=time-JLAST(J)
RLOGG read current
update A-hrs
ISTOP=Ø
write A-hrs passed in BUF storage area
line printer cell...,A-hrs=... on ds.,cycle...
CP1
is this a new mode A(8,J)=Ø)?
N
Y
CUR(J)=.2
JLAST(J)=time
is the cell controlling? (J>A(11,J))
RETURN
Y
SET controller in charge
IMPOSE .2A current
CP2
2 cells in series?
N
Y
RLOGG measure the voltage of the 2 cells and compute DELTAV
DELTAV ≤-10mV?
N
Y
DELTAV ≥10mV?
N
Y
higher cell number?
N
Y
A-hrs exceeded?
N
Y
ISTOP=1
RETURN
CP3

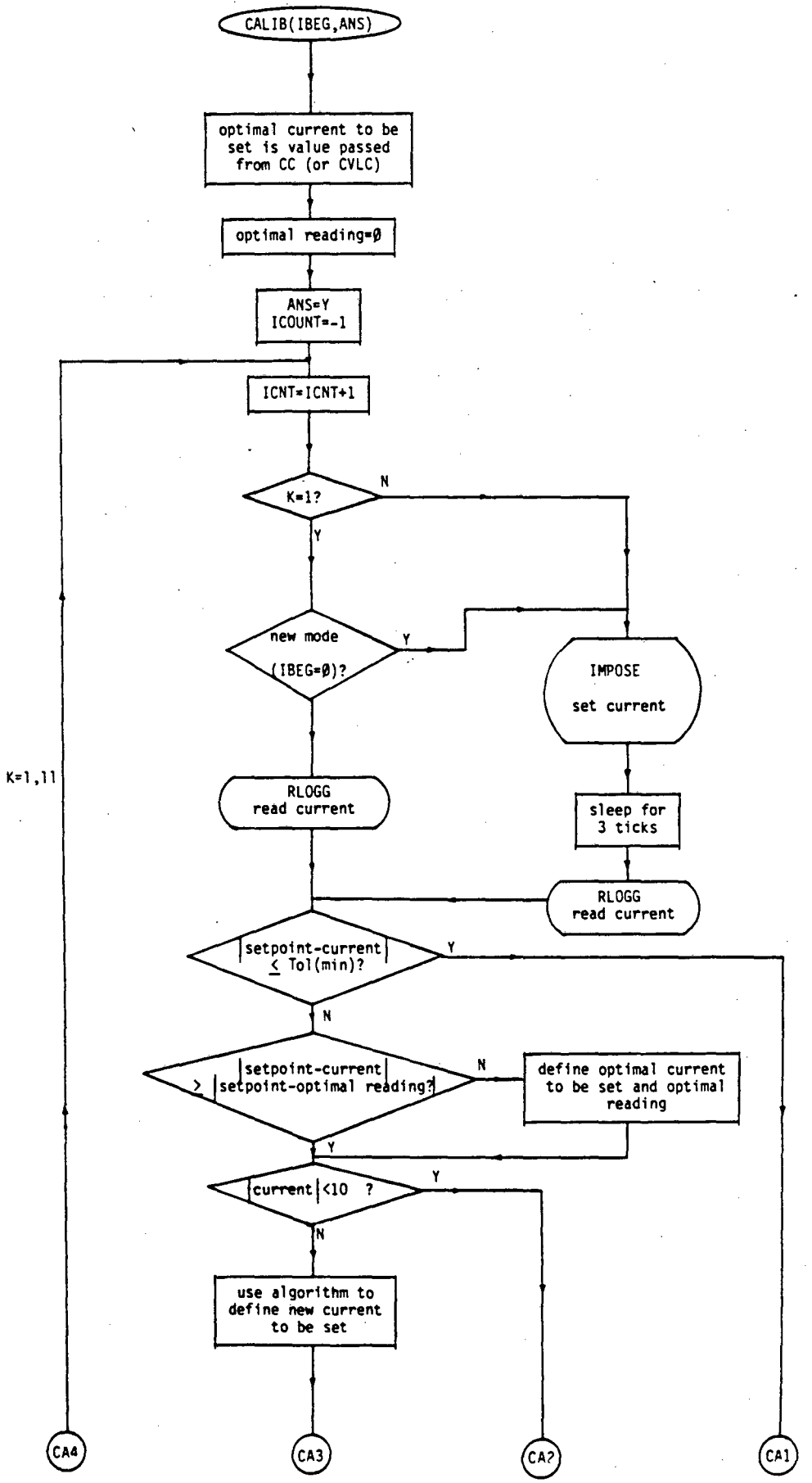
CP-2

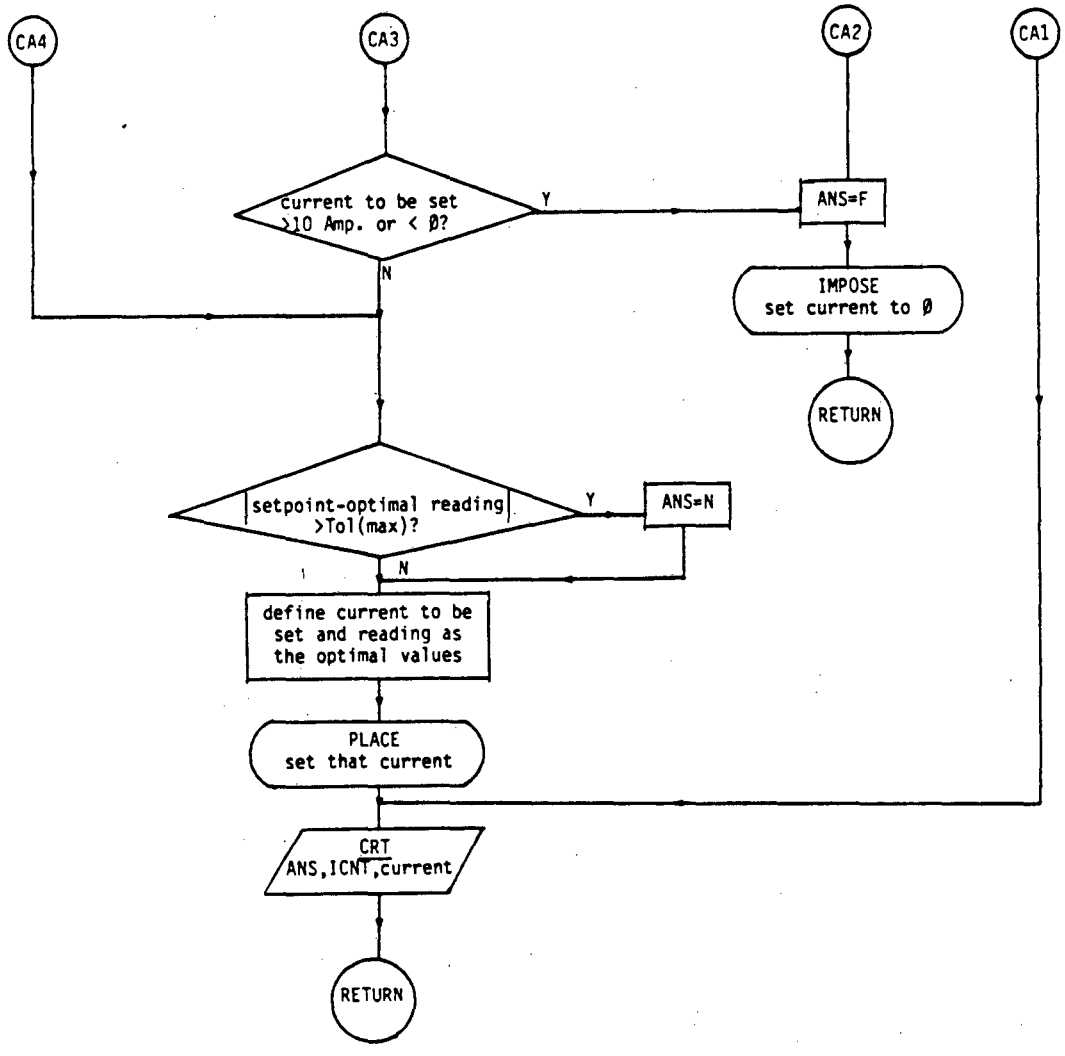

CP3
JDIF=time-JLAST(J)
RLOGG
read current
update A-hrs
is there companion cell?
N
Y
update A-hrs
CRT
A-hrs=...
JLAST of companion= time
JLAST(J)=time
A-hrs exceeded?
N
Y
ISTOP=1
RETURN
CP2
sleep for 3 ticks
SET controller in discharge
IMPOSE .2A current
sleep for 3 ticks
define setpoint for the power
CALIBP(ANS)
ANS=F?
Y
N
set termination flag
Y
is there a companion cell?
Y
N
set termination flag
2 cells in series?
N
RETURN
Y
update CUR for companion
RETURN
CP1
2 cells in series?
N
Y
is companion cell already in open-circuit?
Y
N
CHANGE-action timer for companion cell=NOW!
initialize A-hrs for next half-cycle
RETURN

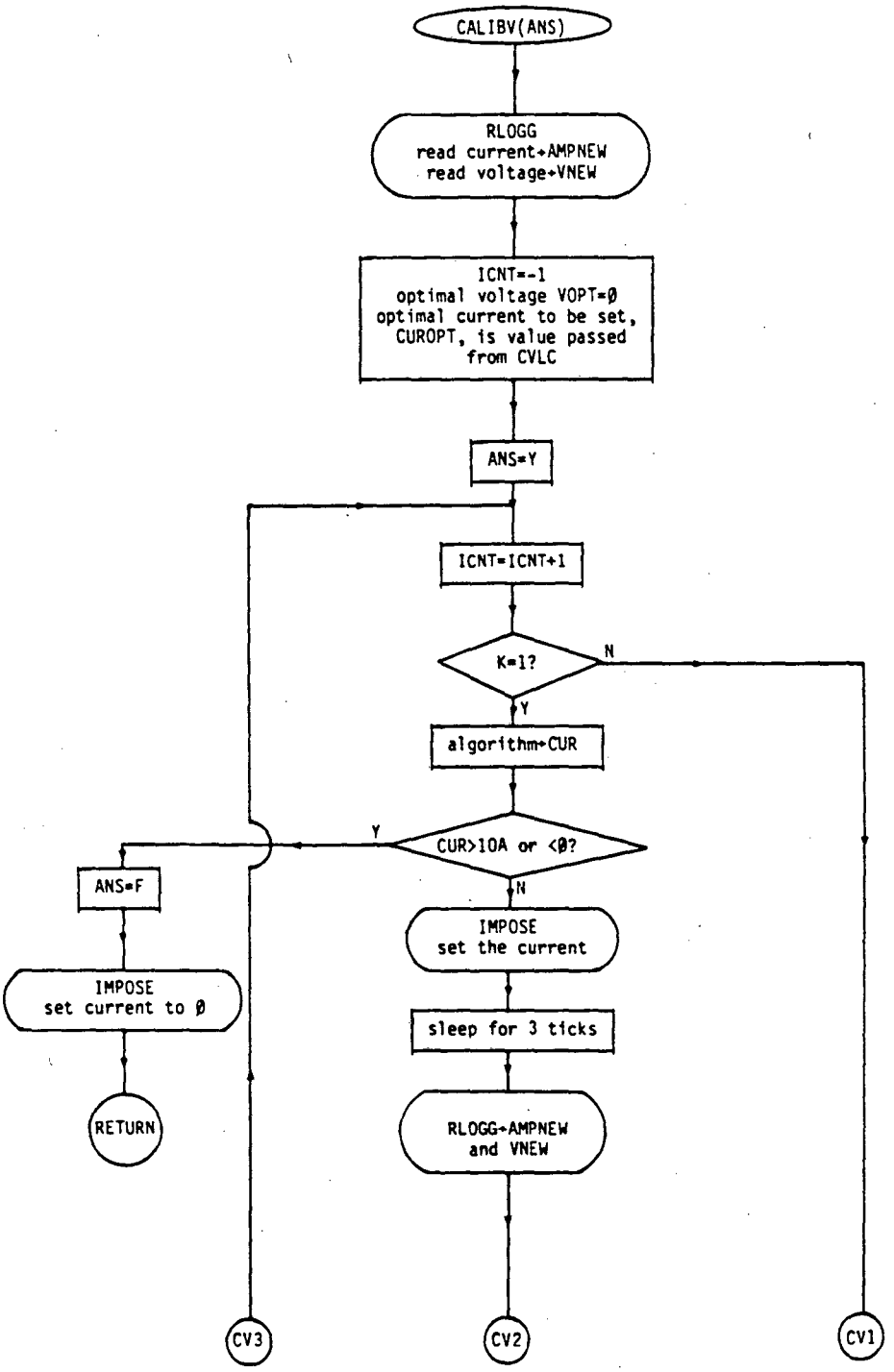
EXCON-1

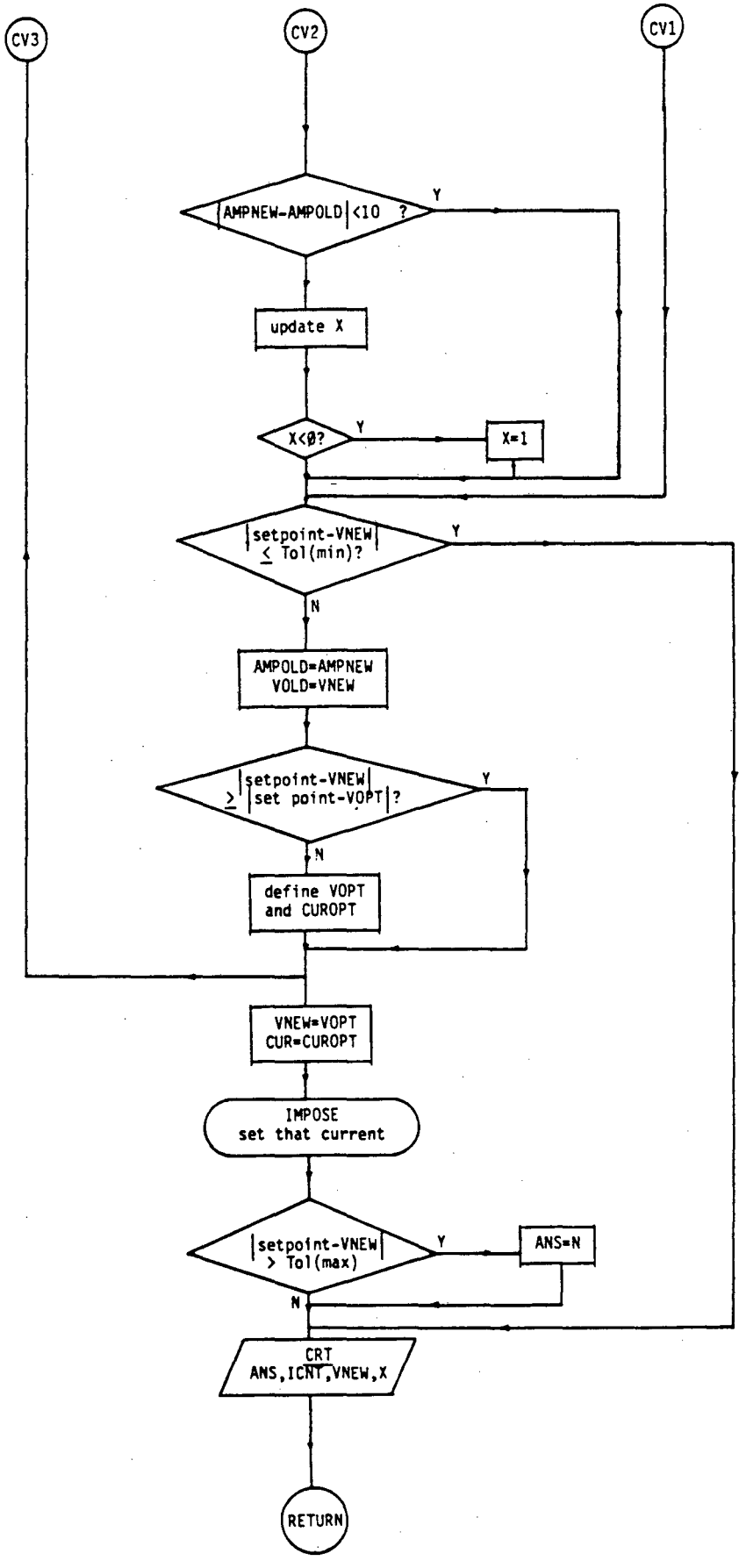

EXCON(ISEC,ISTOP)
get time of day
ISEC=1?
Y
N
single charge or discharge?
charge mode?
define A-hrs to pass
ISEC=2?
is there a companion cell?
is the companion cell in open-circuit?
CHANGE- action timer for companion cell=NOW!
RETURN
start of a new mode? (A(8,J)=0)
update action timer with 2 minutes and 30 seconds
define current A/D channel as output of averager circuit
is this cell controlling? (J>A(11,J))
EXT set current controller in external mode
JLAST(J)=time
RETURN
update action timer with JT(8,J)
is the cell controlling? (J>A(11,J))
JDIF=time-JLAST(J)
E4
E3
ISTOP=1?
JDIF=time-JLAST(J)
RLOGG read current
update A-hrs
ISTOP=0
write A-hrs passed in BUF storage area
line-printer cell...,A-hrs=..., on ch./ds.,cycle...
restore regular current A/D channel
2 cells in series?
is companion cell already in open-circuit?
E2
E1

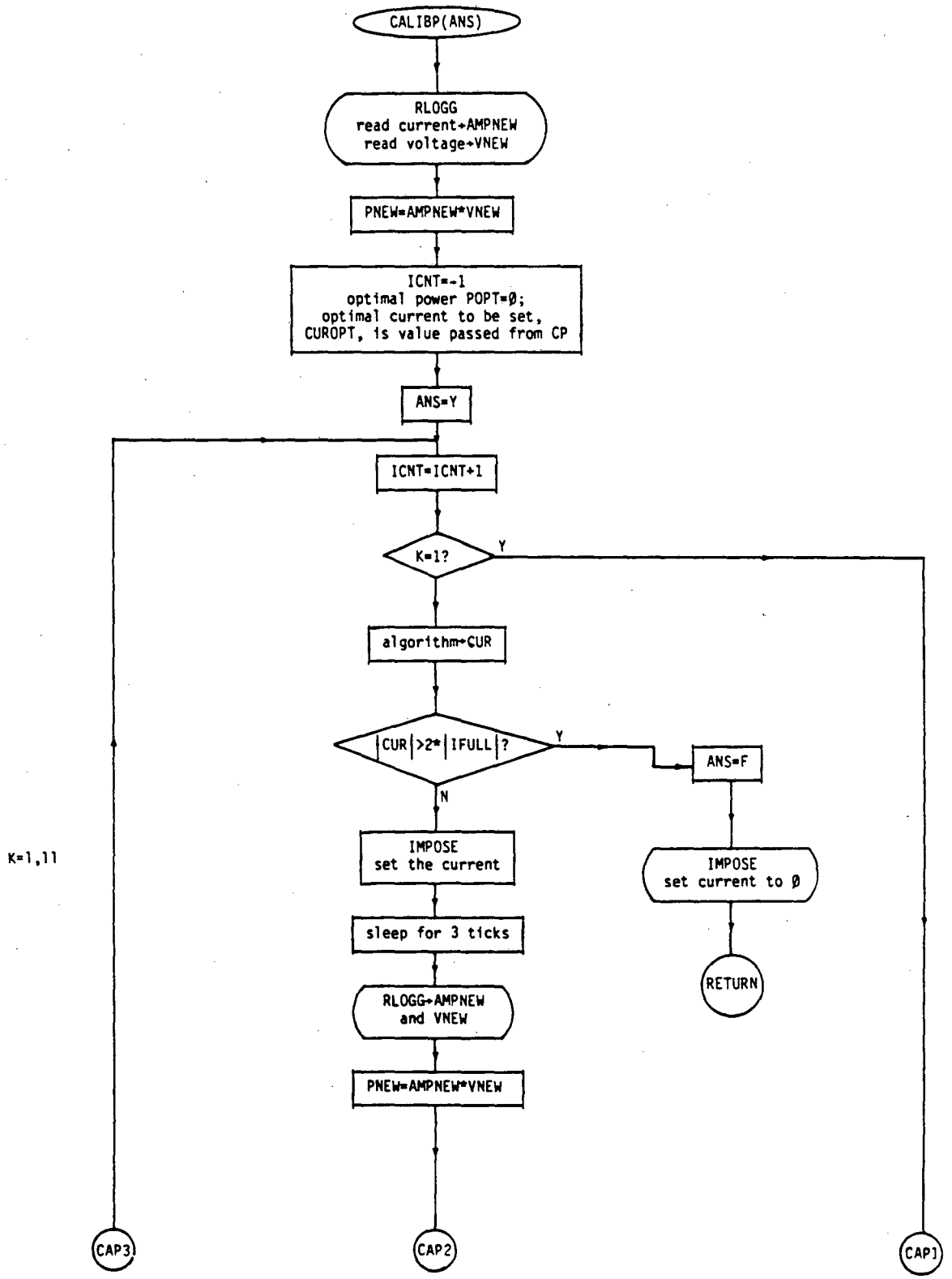

E4
E3
E2
E1
RLOGG
read current
A-hrs=A-hrs+JDIF*current
CRT
A-hrs=...
is there a companion cell?
N
Y
update its A-hrs
CRT
A-hrs=...
JLAST of companion =time
JLAST(J)=time
RETURN
N
A-hrs exceeded?
Y
ISTOP=1
RETURN
CHANGE-action timer for companion cell=NOW!
initialize A-hrs for next half-cycle
single charge or discharge?
Y
N
RETURN
was the cell on charge (A(3,J)=1)?
N
Y
is cell being put out of operation?
N
Y
for re-start purposes... (see CC)
RETURN

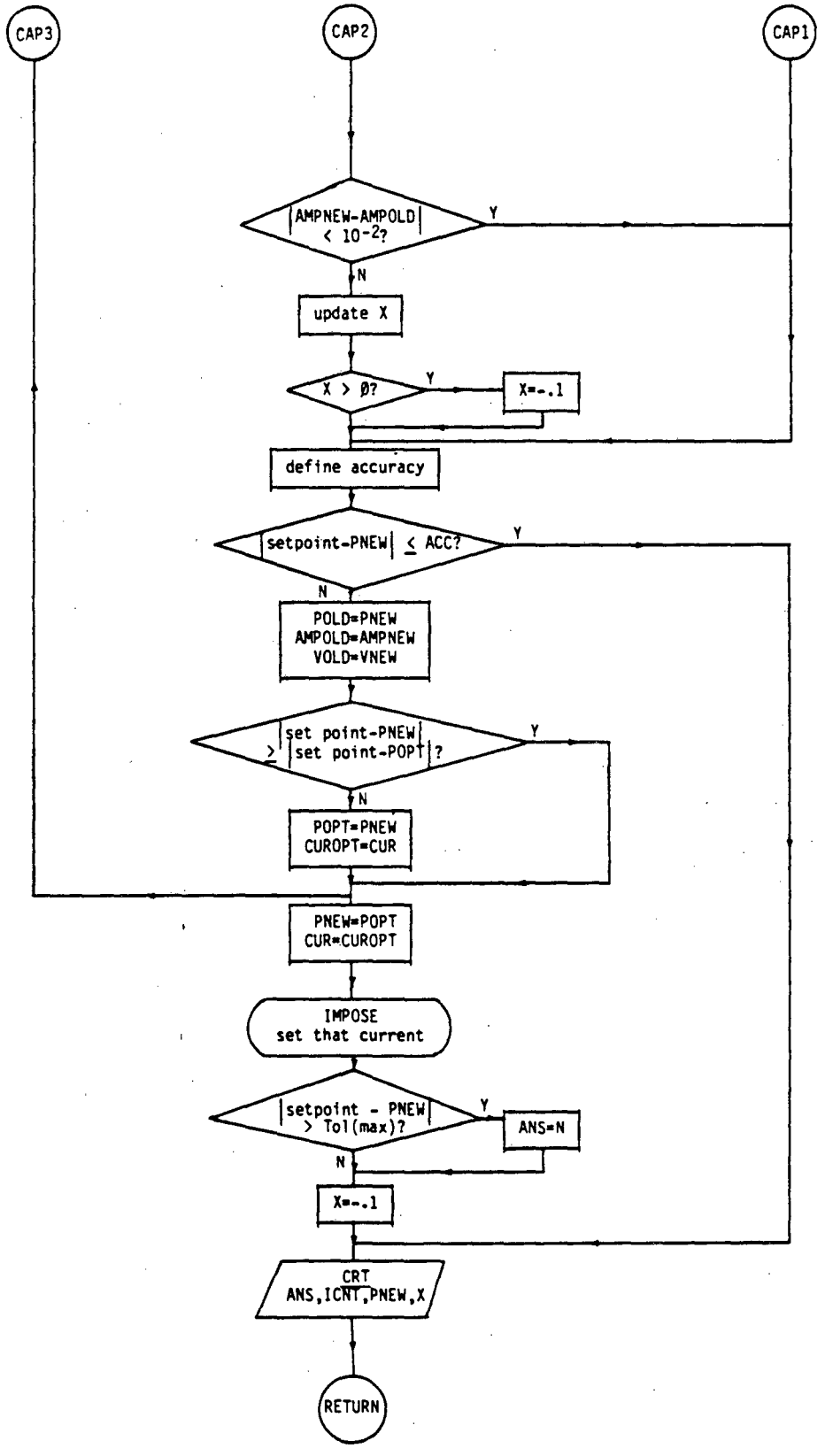

CALIB(IBEG,ANS)
optimal current to be set is value passed from CC (or CVLC)
optimal reading=Ø
ANS=Y
ICOUNT=-1
ICNT=ICNT+1
K=1?
N
Y
new mode (IBEG=Ø)?
Y
IMPOSE set current
K=1,11
RLOGG read current
sleep for 3 ticks
RLOGG read current
|setpoint-current| ≤ Tol(min)?
Y
N
|setpoint-current| > |setpoint-optimal reading|?
N
define optimal current to be set and optimal reading
Y
|current| <10 ?
Y
N
use algorithm to define new current to be set
CA4
CA3
CA2
CA1


CA4
CA3
CA2
CA1
current to be set >10 Amp. or < Ø?
Y
N
ANS=F
IMPOSE set current to Ø
RETURN
setpoint-optimal reading >Tol(max)?
Y
ANS=N
N
define current to be set and reading as the optimal values
PLACE set that current
CRT ANS,ICNT,current
RETURN

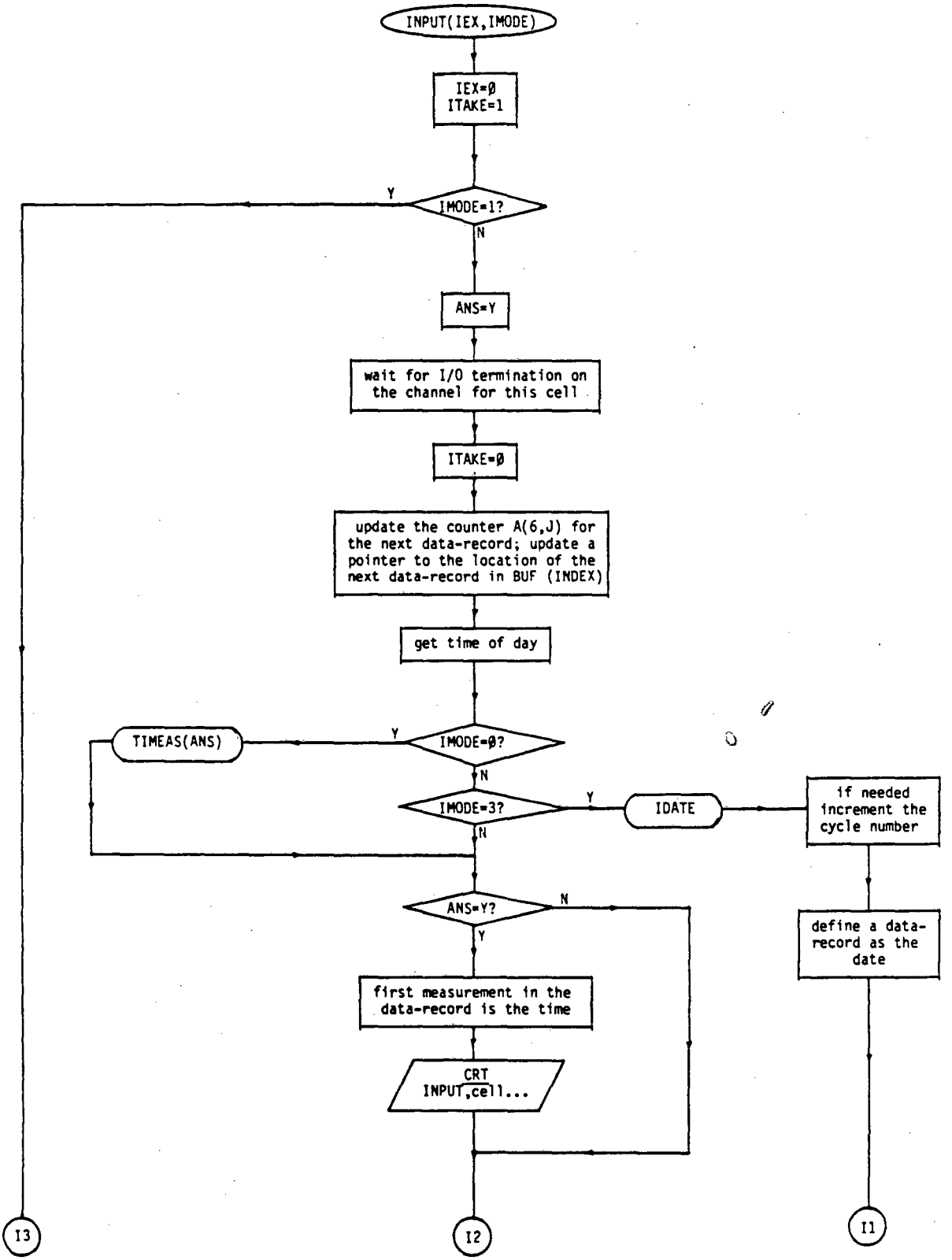

CALIBV(ANS)
RLOGG
read current→AMPNEW
read voltage→VNEW
ICNT=-1
optimal voltage VOPT=Ø
optimal current to be set,
CUROPT, is value passed
from CVLC
ANS=Y
ICNT=ICNT+1
K=1?
N
Y
algorithm→CUR
CUR>10A or <Ø?
Y
N
ANS=F
IMPOSE
set current to Ø
IMPOSE
set the current
sleep for 3 ticks
RLOGG→AMPNEW
and VNEW
RETURN
CV3
CV2
CV1

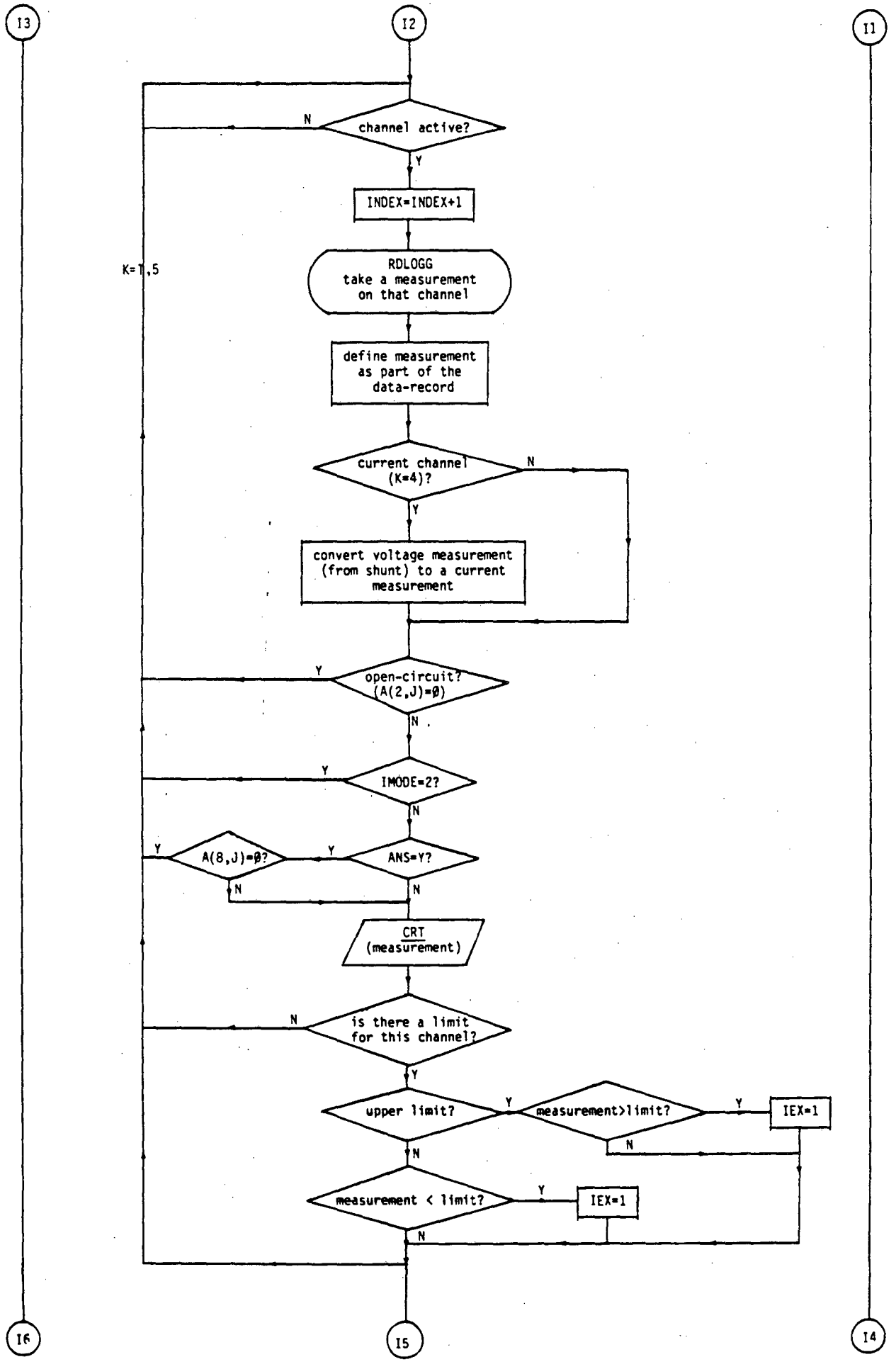

CV3
CV2
CV1
|AMPNEW-AMPOLD| <10 ?
Y
update X
X<0?
Y
X=1
|setpoint-VNEW| ≤ Tol(min)?
Y
N
AMPOLD=AMPNEW
VOLD=VNEW
≥ |setpoint-VNEW| |set point-VOPT| ?
Y
N
define VOPT and CUROPT
VNEW=VOPT
CUR=CUROPT
IMPOSE set that current
|setpoint-VNEW| > Tol(max)
Y
ANS=N
N
CRT ANS,ICNT,VNEW,X
RETURN

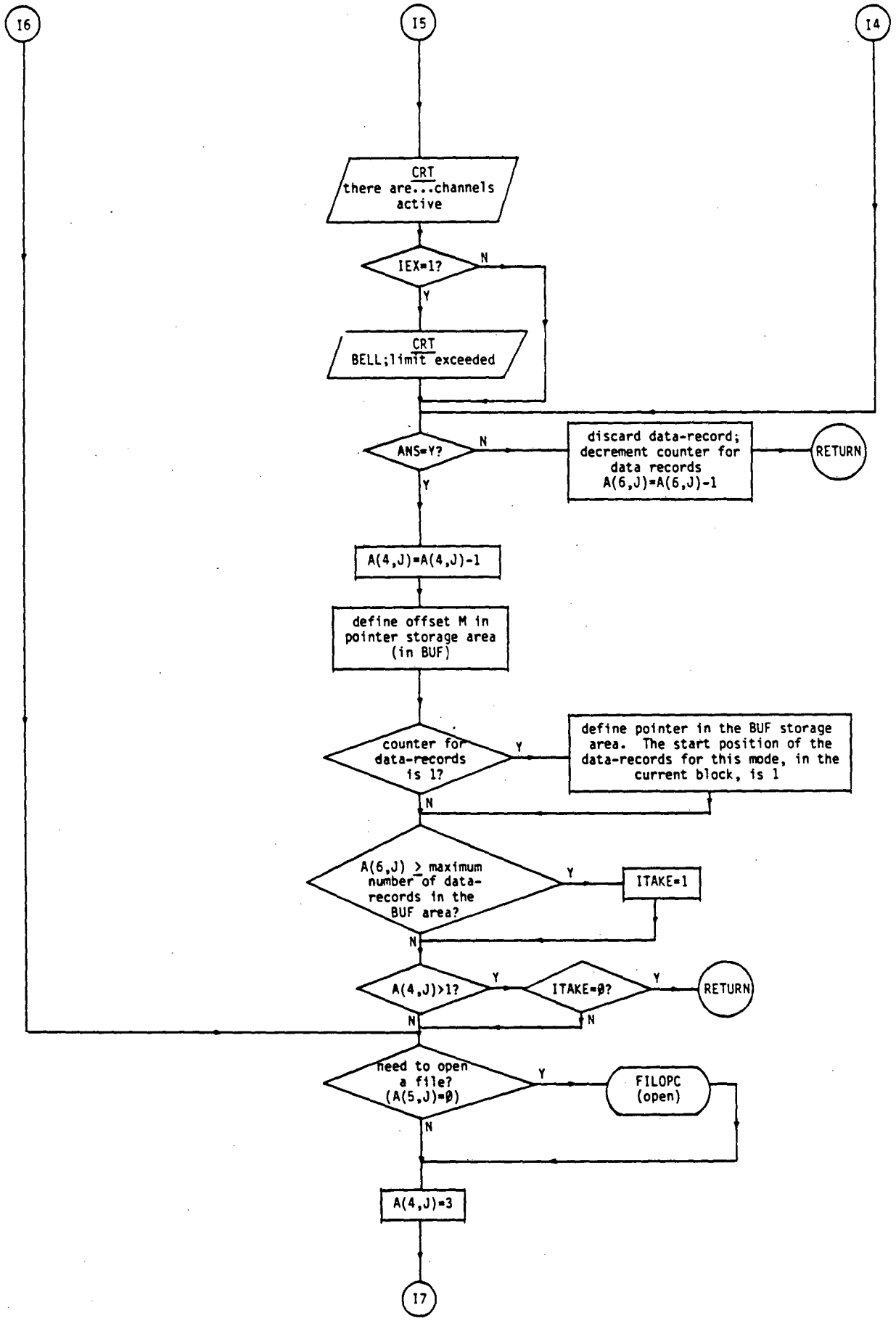

CALIBP(ANS)
RLOGG
read current→AMPNEW
read voltage→VNEW
PNEW=AMPNEW*VNEW
ICNT=-1
optimal power POPT=Ø;
optimal current to be set,
CUROPT, is value passed from CP
ANS=Y
ICNT=ICNT+1
K=1?
Y
algorithm→CUR
|CUR|>2*|IFULL|?
Y
ANS=F
N
IMPOSE
set the current
IMPOSE
set current to Ø
sleep for 3 ticks
RETURN
RLOGG→AMPNEW
and VNEW
PNEW=AMPNEW*VNEW
K=1,11
CAP3
CAP2
CAP1

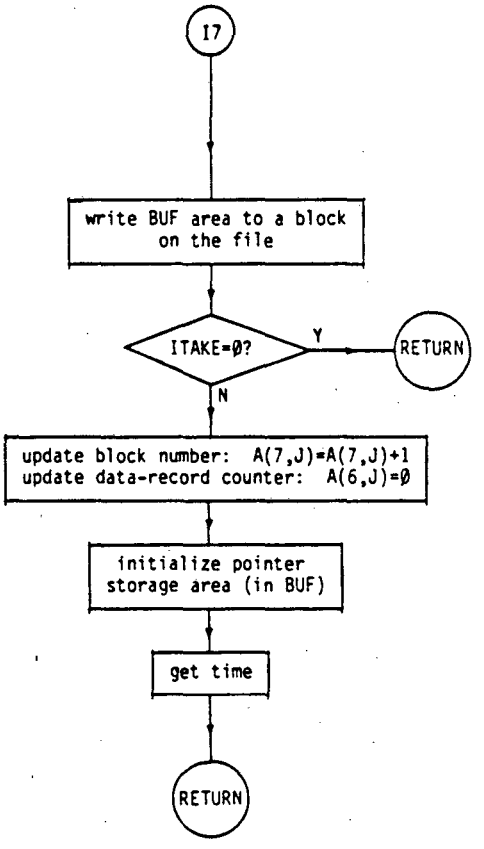

CAP3
CAP2
CAP1
|AMPNEW-AMPOLD| < 10^-2?
Y
N
update X
X > 0?
Y
X=-.1
define accuracy
|setpoint-PNEW| ≤ ACC?
Y
N
POLD=PNEW
AMPOLD=AMPNEW
VOLD=VNEW
|set point-PNEW| ≥ |set point-POPT|?
Y
N
POPT=PNEW
CUROPT=CUR
PNEW=POPT
CUR=CUROPT
IMPOSE
set that current
|setpoint - PNEW| > Tol(max)?
Y
ANS=N
N
X=-.1
CRT
ANS,ICNT,PNEW,X
RETURN

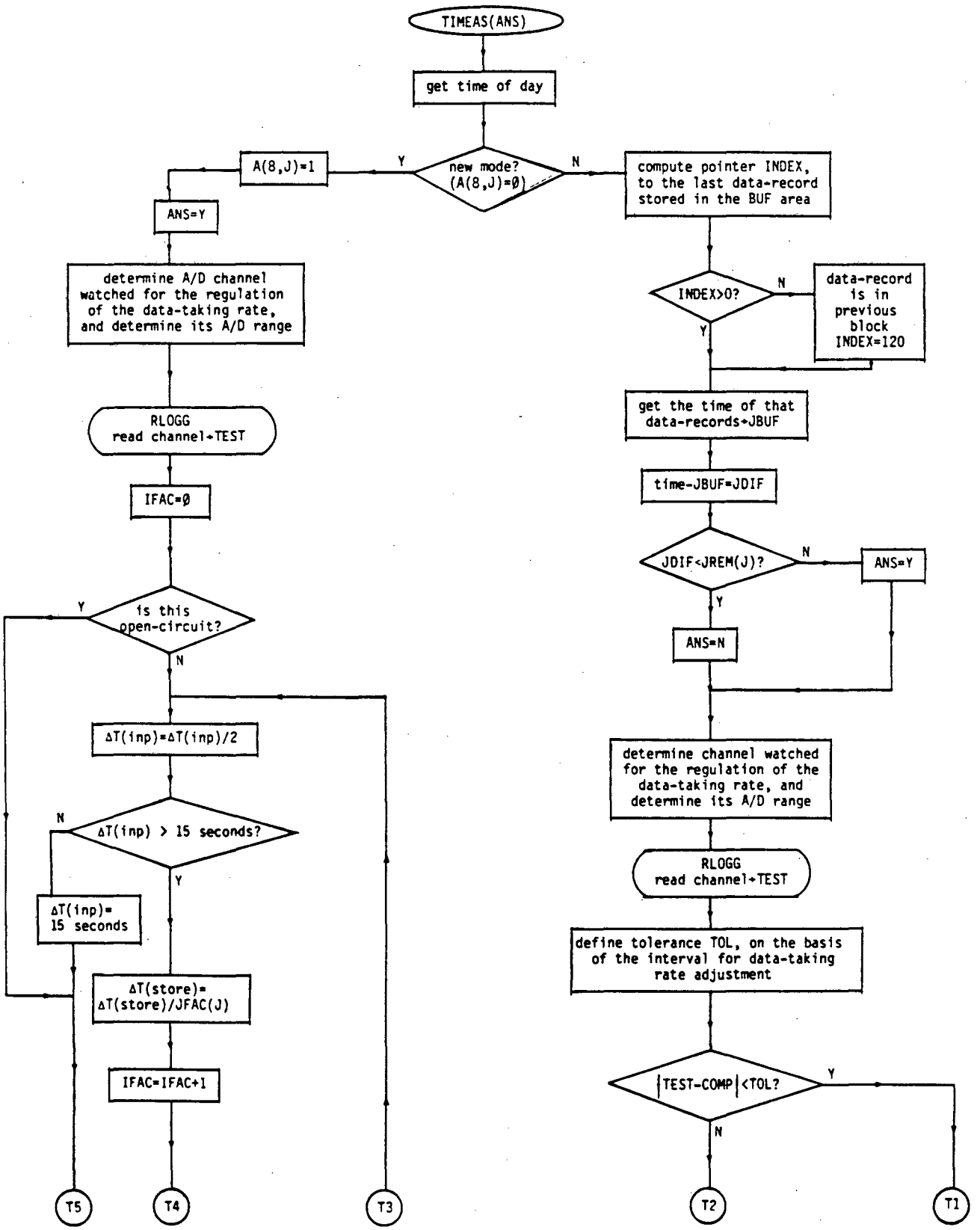
INPUT-1

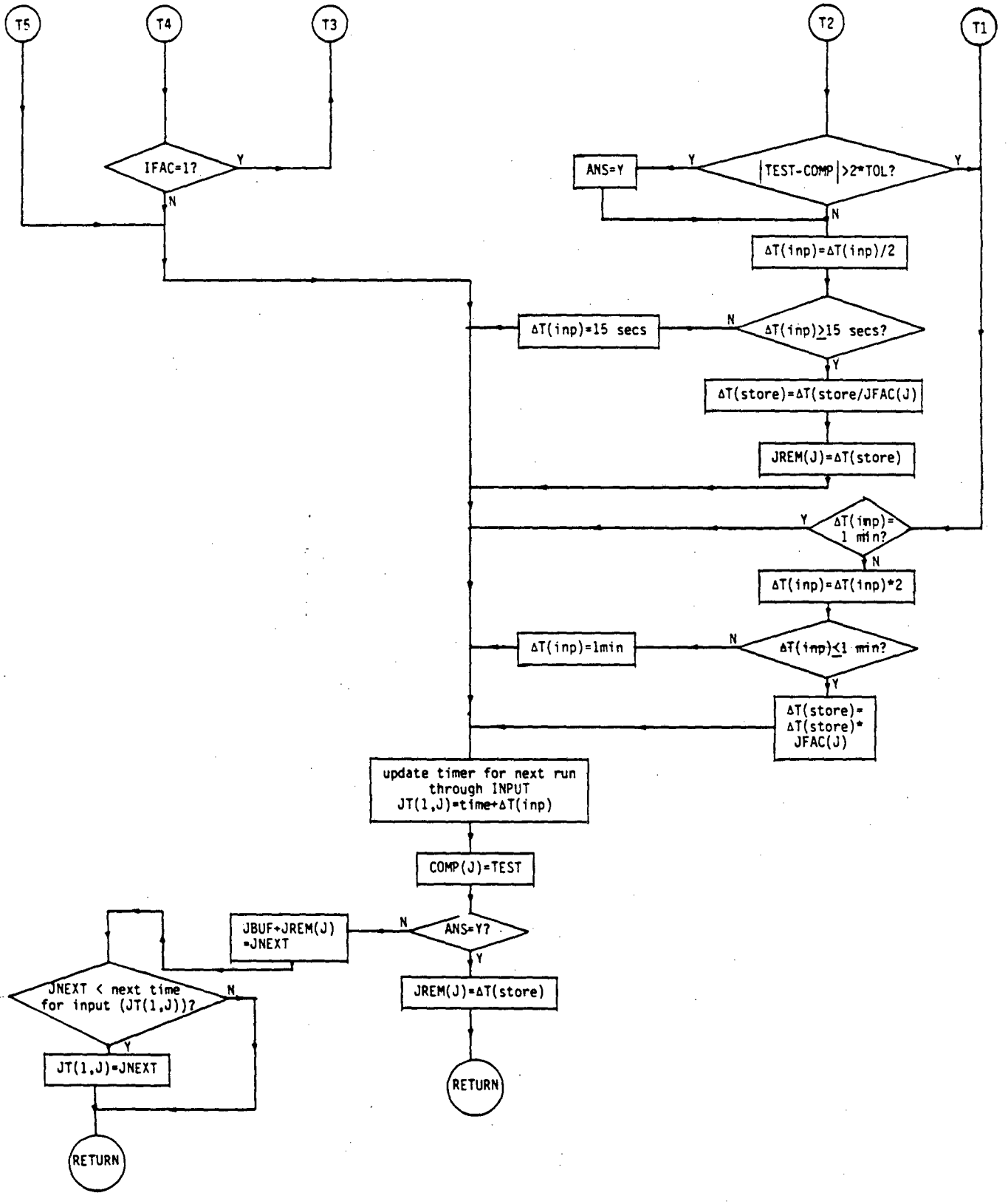

INPUT(IEX,IMODE)
IEX=0
ITAKE=1
IMODE=1?
Y
N
ANS=Y
wait for I/O termination on the channel for this cell
ITAKE=0
update the counter A(6,J) for the next data-record; update a pointer to the location of the next data-record in BUF (INDEX)
get time of day
IMODE=0?
TIMEAS(ANS)
IMODE=3?
IDATE
if needed increment the cycle number
define a data-record as the date
ANS=Y?
first measurement in the data-record is the time
CRT INPUT,cell...
I1
I2
I3

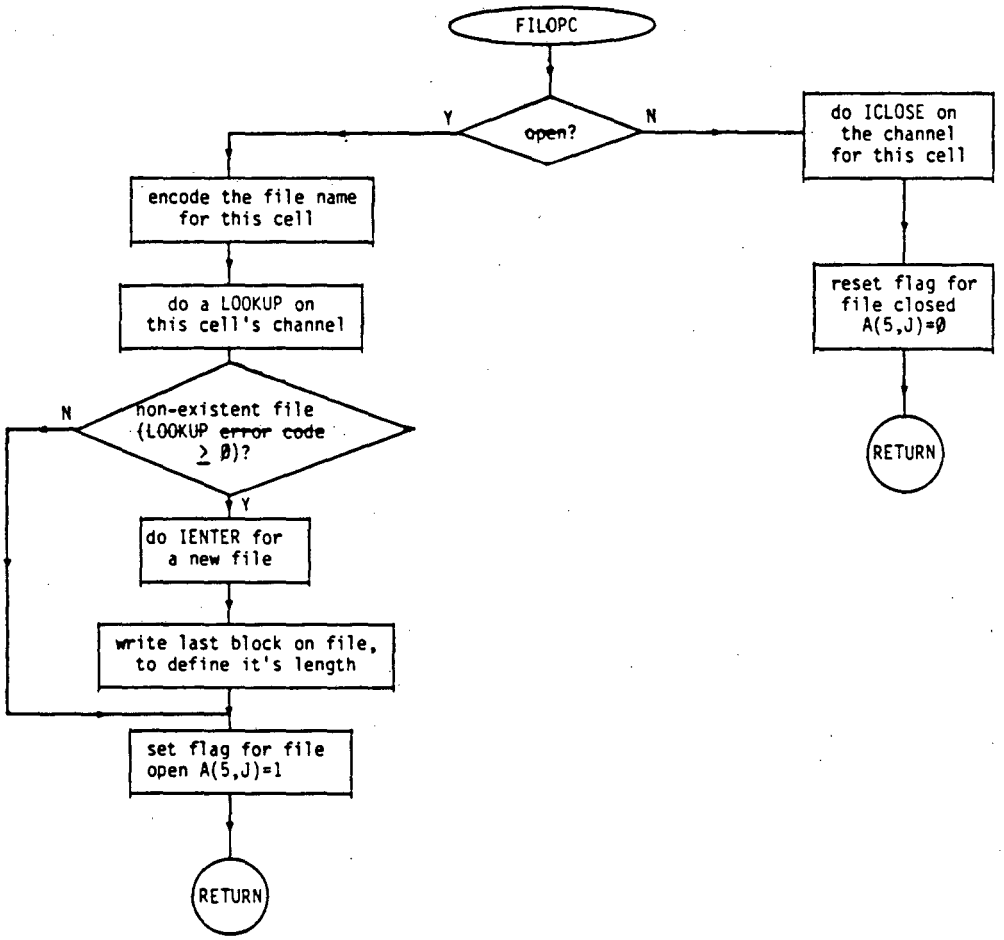

I3
I2
I1
channel active?
N
Y
INDEX=INDEX+1
K=1,5
RDLOGG
take a measurement
on that channel
define measurement
as part of the
data-record
current channel
(K=4)?
N
Y
convert voltage measurement
(from shunt) to a current
measurement
open-circuit?
(A(2,J)=0)
Y
N
IMODE=2?
Y
N
A(8,J)=0?
Y
N
ANS=Y?
Y
N
CRT
(measurement)
is there a limit
for this channel?
N
Y
upper limit?
Y
measurement>limit?
Y
IEX=1
N
N
measurement < limit?
Y
IEX=1
N
I6
I5
I4

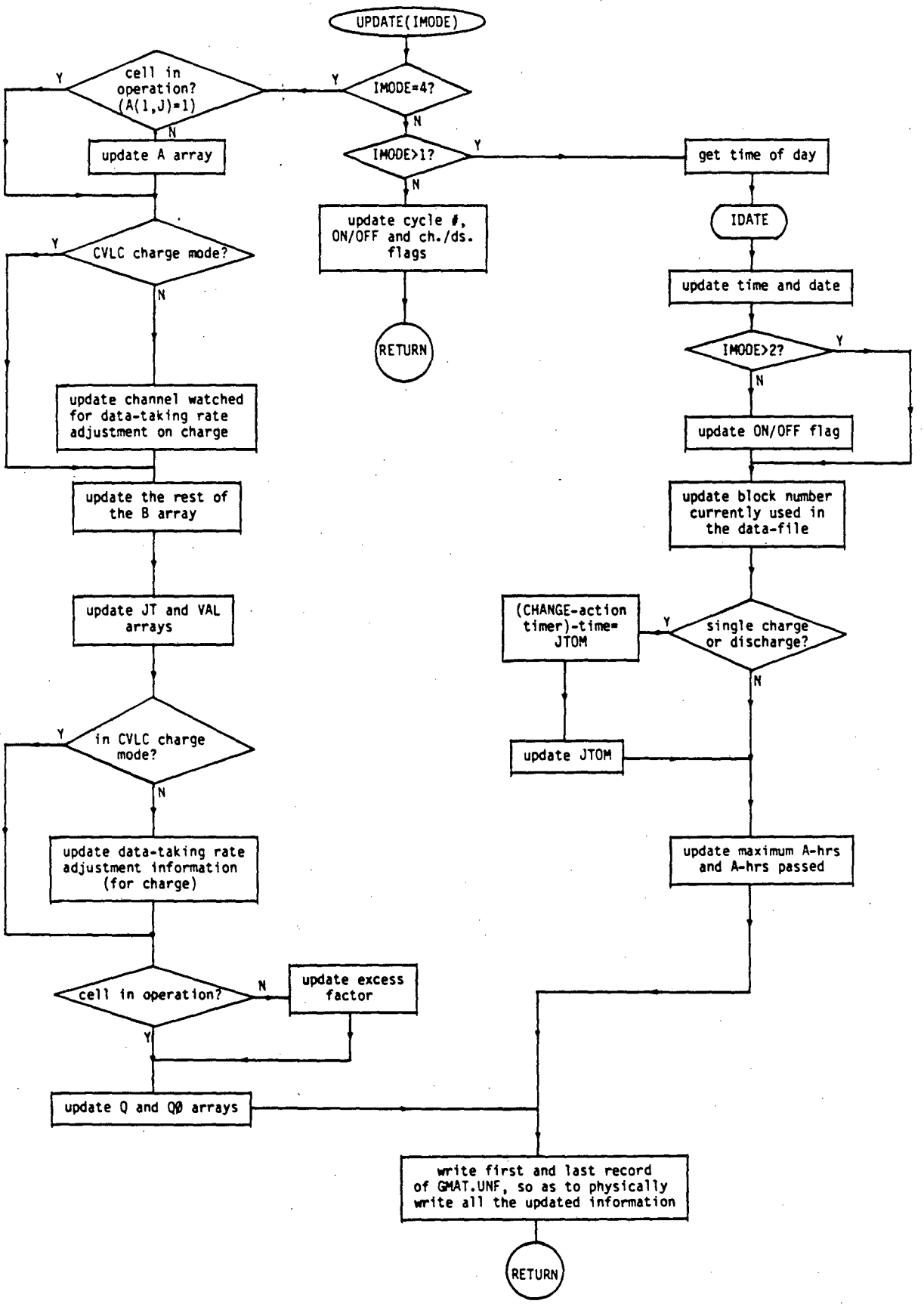

I6
I5
I4
CRT
there are...channels
active
IEX=1?
N
Y
CRT
BELL;limit exceeded
ANS=Y?
N
discard data-record;
decrement counter for
data records
A(6,J)=A(6,J)-1
RETURN
Y
A(4,J)=A(4,J)-1
define offset M in
pointer storage area
(in BUF)
counter for
data-records
is 1?
Y
define pointer in the BUF storage
area. The start position of the
data-records for this mode, in the
current block, is 1
N
A(6,J) ≥ maximum
number of data-
records in the
BUF area?
Y
ITAKE=1
N
A(4,J)>1?
Y
ITAKE=Ø?
Y
RETURN
N
N
need to open
a file?
(A(5,J)=Ø)
Y
FILOPC
(open)
N
A(4,J)=3
I7


I7
write BUF area to a block on the file
ITAKE=Ø?
Y
RETURN
N
update block number: A(7,J)=A(7,J)+1
update data-record counter: A(6,J)=Ø
initialize pointer storage area (in BUF)
get time
RETURN

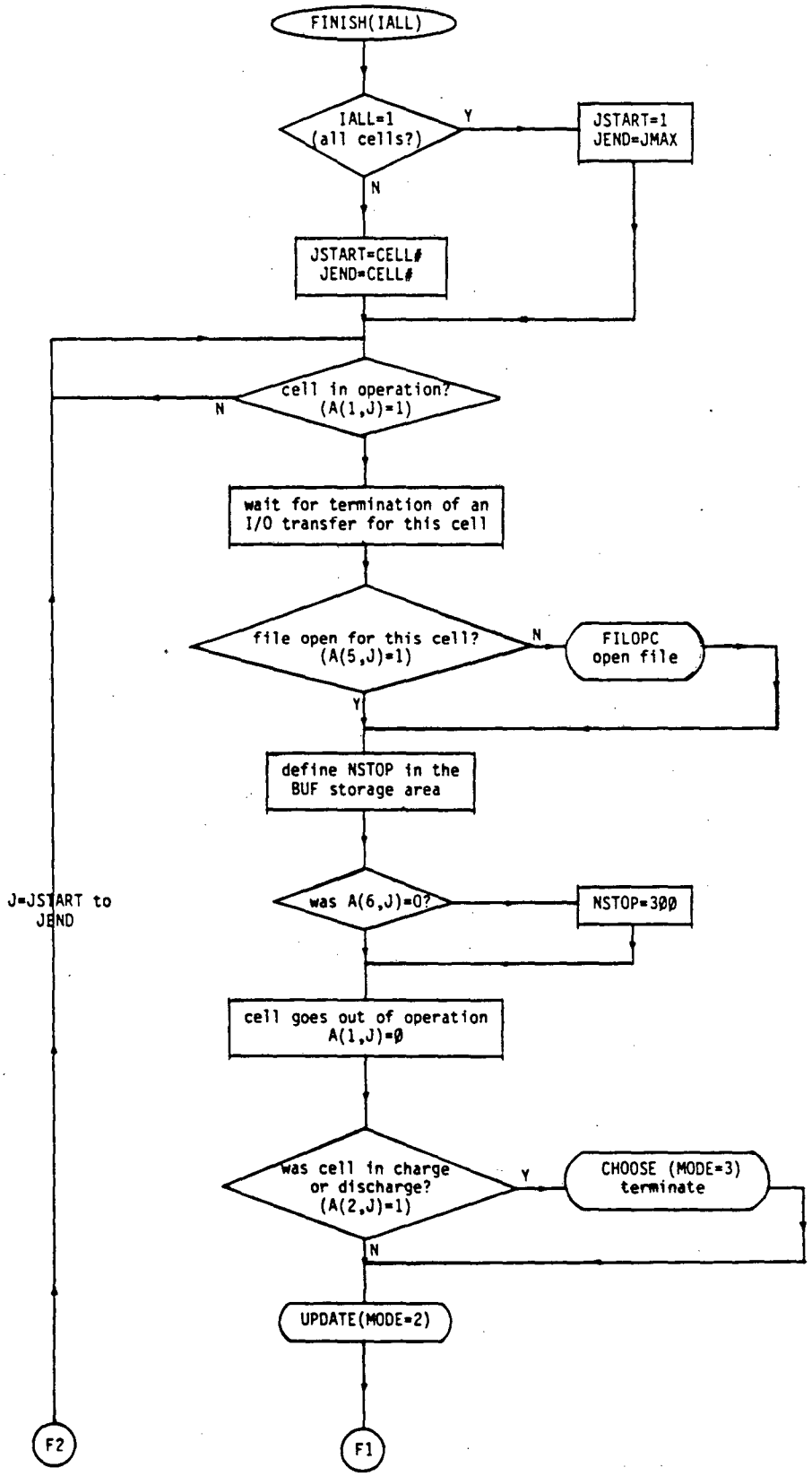

TIMEAS(ANS)
get time of day
new mode? (A(8,J)=Ø)
Y
N
A(8,J)=1
ANS=Y
determine A/D channel watched for the regulation of the data-taking rate, and determine its A/D range
RLOGG read channel→TEST
IFAC=Ø
is this open-circuit?
Y
N
ΔT(inp)=ΔT(inp)/2
ΔT(inp) > 15 seconds?
N
Y
ΔT(inp)= 15 seconds
ΔT(store)= ΔT(store)/JFAC(J)
IFAC=IFAC+1
T5
T4
T3
compute pointer INDEX, to the last data-record stored in the BUF area
INDEX>0?
N
Y
data-record is in previous block INDEX=120
get the time of that data-records→JBUF
time-JBUF=JDIF
JDIF<JREM(J)?
N
Y
ANS=Y
ANS=N
determine channel watched for the regulation of the data-taking rate, and determine its A/D range
RLOGG read channel→TEST
define tolerance TOL, on the basis of the interval for data-taking rate adjustment
|TEST-COMP|<TOL?
Y
N
T2
T1


T5
T4
T3
T2
T1
IFAC=1?
Y
N
ANS=Y
Y
|TEST-COMP|>2*TOL?
Y
N
ΔT(inp)=ΔT(inp)/2
ΔT(inp)=15 secs
N
ΔT(inp)≥15 secs?
Y
ΔT(store)=ΔT(store/JFAC(J)
JREM(J)=ΔT(store)
Y
ΔT(inp)=
1 min?
N
ΔT(inp)=ΔT(inp)*2
ΔT(inp)=1min
N
ΔT(inp)≤1 min?
Y
ΔT(store)=
ΔT(store)*
JFAC(J)
update timer for next run
through INPUT
JT(1,J)=time+ΔT(inp)
COMP(J)=TEST
JBUF←JREM(J)
=JNEXT
N
ANS=Y?
Y
JNEXT < next time
for input (JT(1,J))?
N
Y
JT(1,J)=JNEXT
JREM(J)=ΔT(store)
RETURN
RETURN

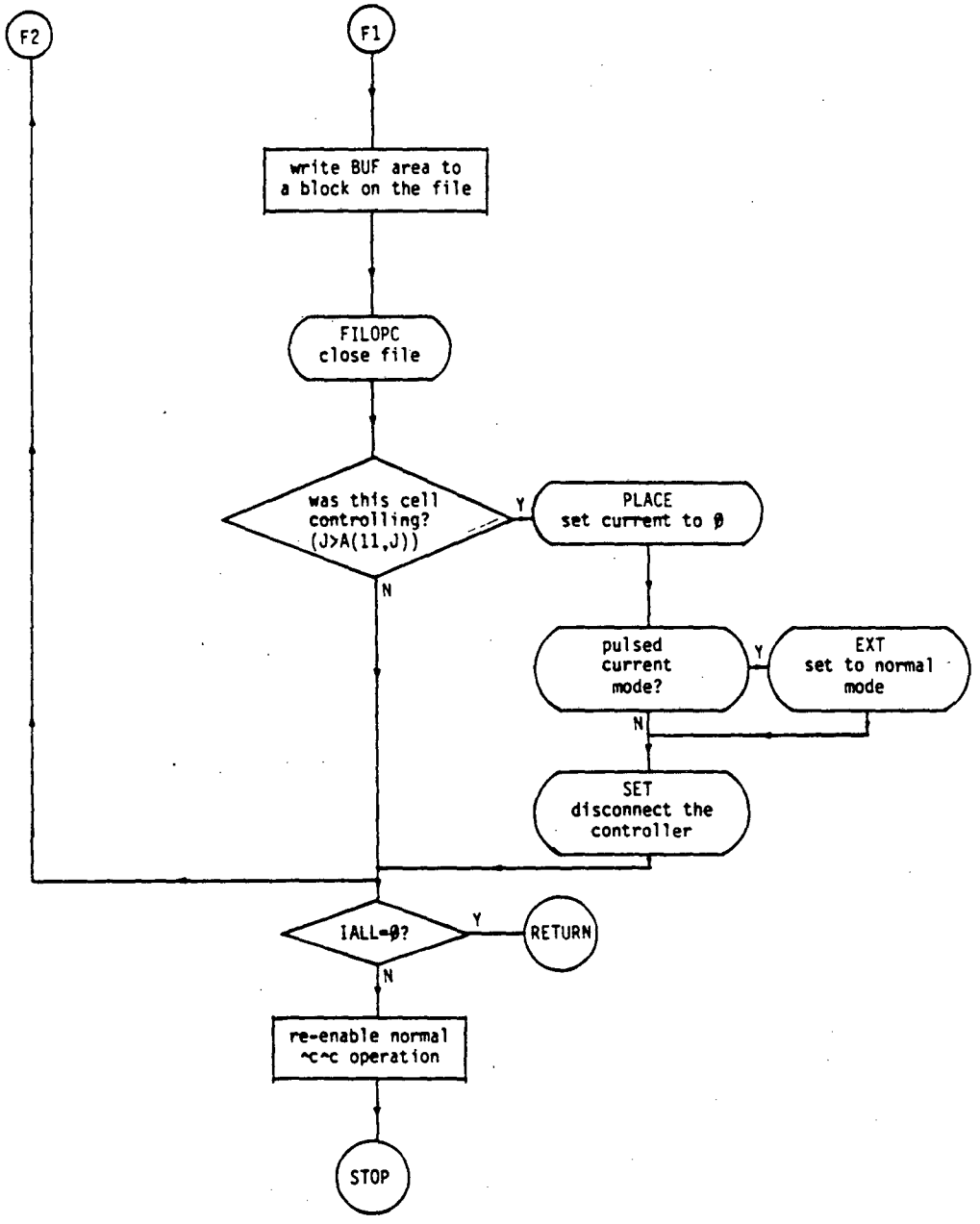

FILOPC
open?
Y
N
do ICLOSE on the channel for this cell
reset flag for file closed A(5,J)=Ø
RETURN
encode the file name for this cell
do a LOOKUP on this cell's channel
non-existent file (LOOKUP error code ≥ Ø)?
N
Y
do IENTER for a new file
write last block on file, to define it's length
set flag for file open A(5,J)=1
RETURN

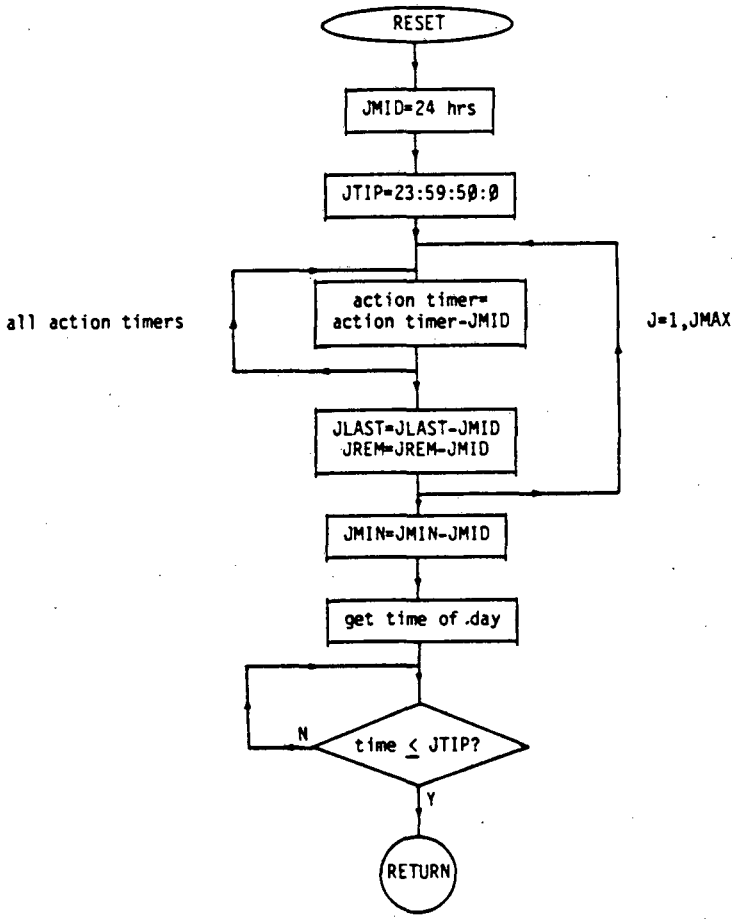

UPDATE(IMODE)
IMODE=4?
Y
N
cell in operation? (A(1,J)=1)
Y
N
update A array
IMODE>1?
Y
N
update cycle #, ON/OFF and ch./ds. flags
RETURN
get time of day
IDATE
update time and date
IMODE>2?
Y
N
update ON/OFF flag
update block number currently used in the data-file
CVLC charge mode?
Y
N
update channel watched for data-taking rate adjustment on charge
update the rest of the B array
update JT and VAL arrays
in CVLC charge mode?
Y
N
update data-taking rate adjustment information (for charge)
(CHANGE-action timer)-time= JTOM
single charge or discharge?
Y
N
update JTOM
update maximum A-hrs and A-hrs passed
cell in operation?
N
Y
update excess factor
update Q and Q0 arrays
write first and last record of GMAT.UNF, so as to physically write all the updated information
RETURN

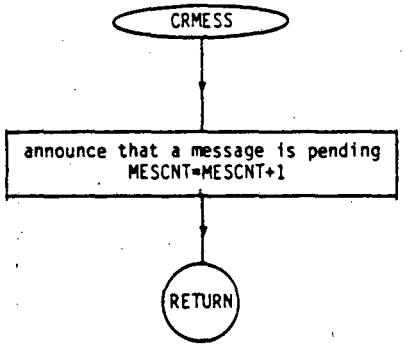
FINISH-1

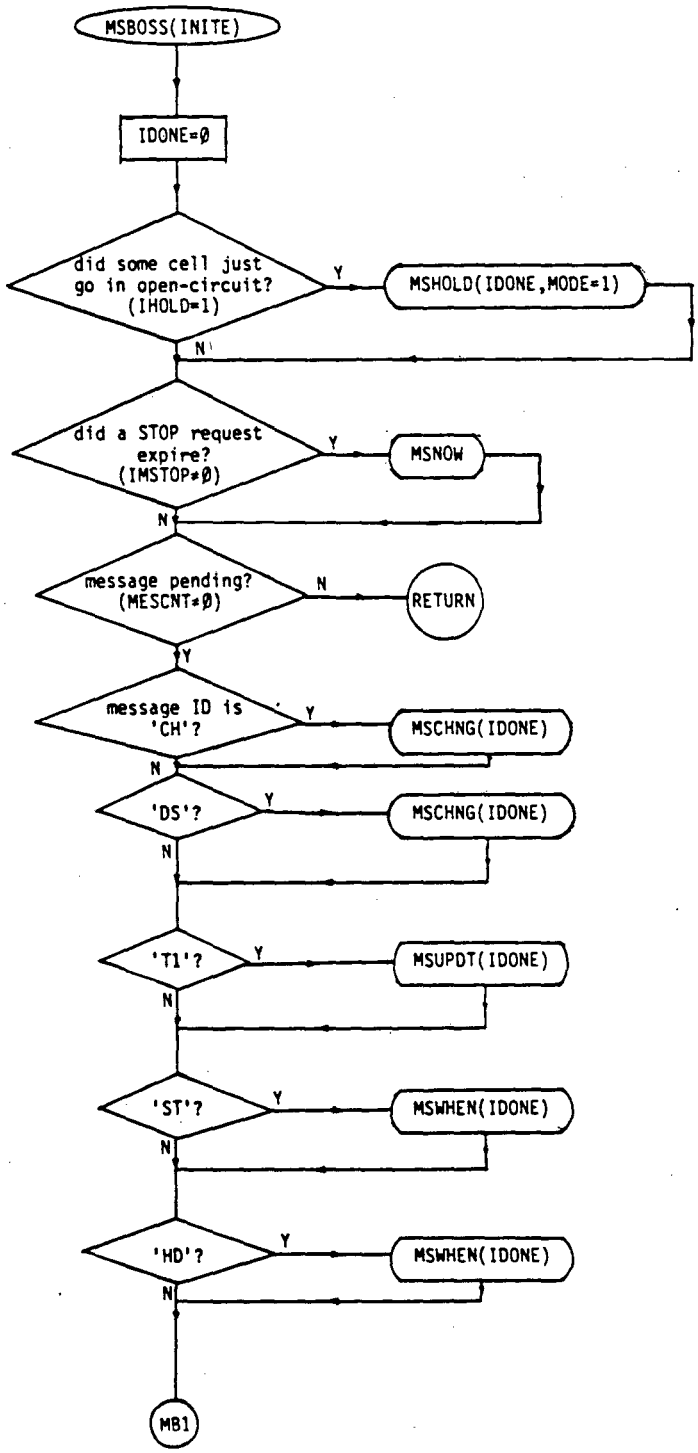

FINISH(IALL)
IALL=1 (all cells?)
Y
JSTART=1 JEND=JMAX
N
JSTART=CELL# JEND=CELL#
cell in operation? (A(1,J)=1)
N
wait for termination of an I/O transfer for this cell
file open for this cell? (A(5,J)=1)
N
FILOPC open file
Y
define NSTOP in the BUF storage area
J=JSTART to JEND
was A(6,J)=0?
NSTOP=300
cell goes out of operation A(1,J)=0
was cell in charge or discharge? (A(2,J)=1)
Y
CHOOSE (MODE=3) terminate
N
UPDATE(MODE=2)
F2
F1

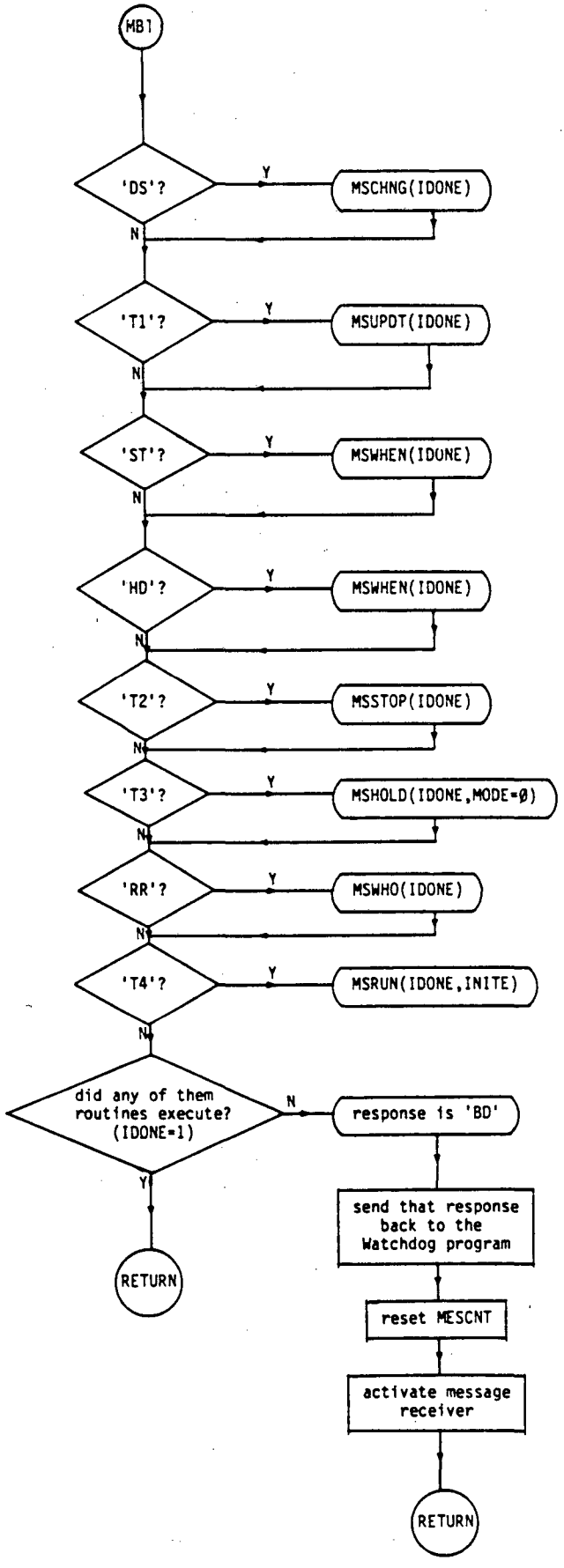
FINISH-2

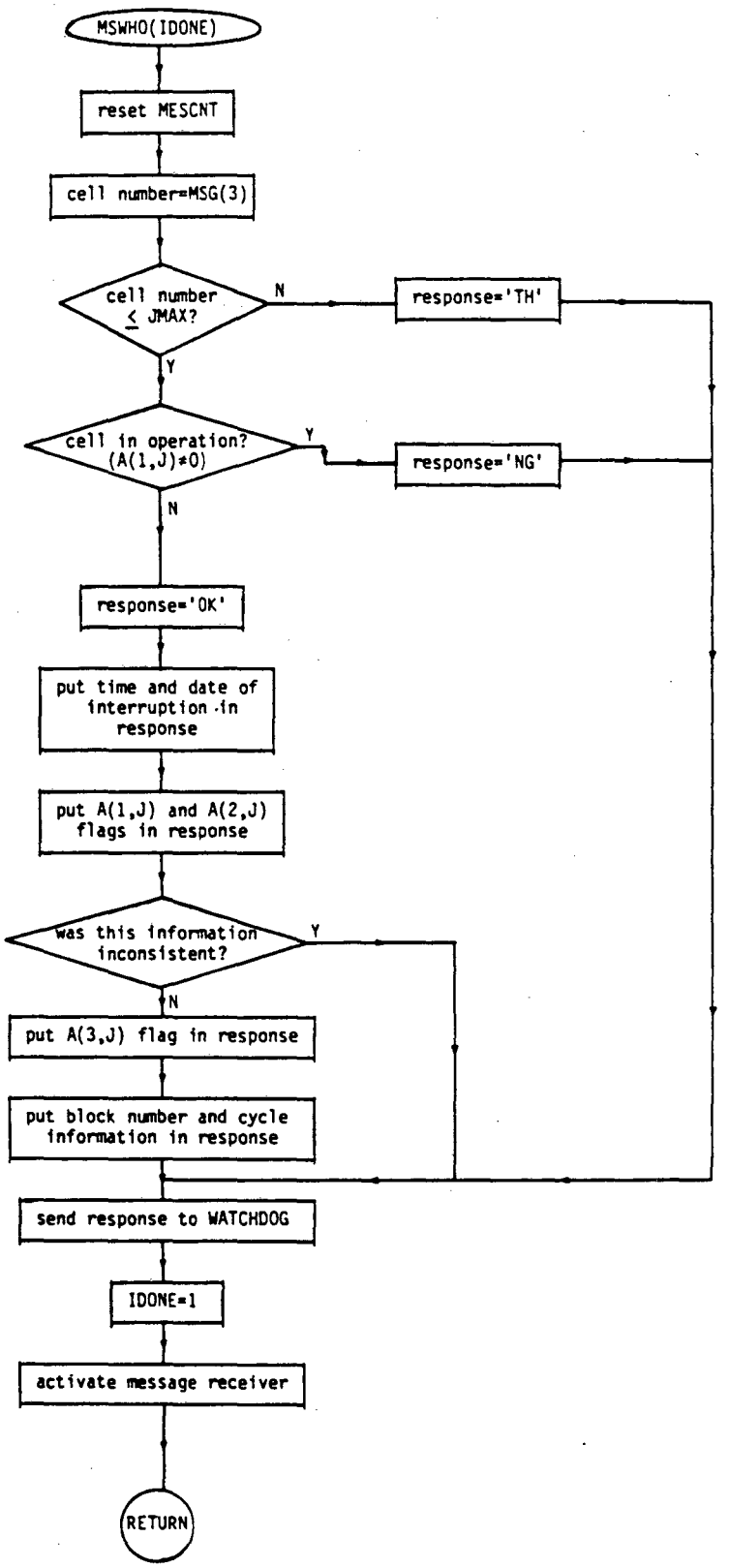

F2
F1
write BUF area to a block on the file
FILOPC close file
was this cell controlling? (J>A(11,J))
Y
PLACE set current to ∅
N
pulsed current mode?
Y
EXT set to normal mode
N
SET disconnect the controller
IALL=∅?
Y
RETURN
N
re-enable normal ^c^c operation
STOP

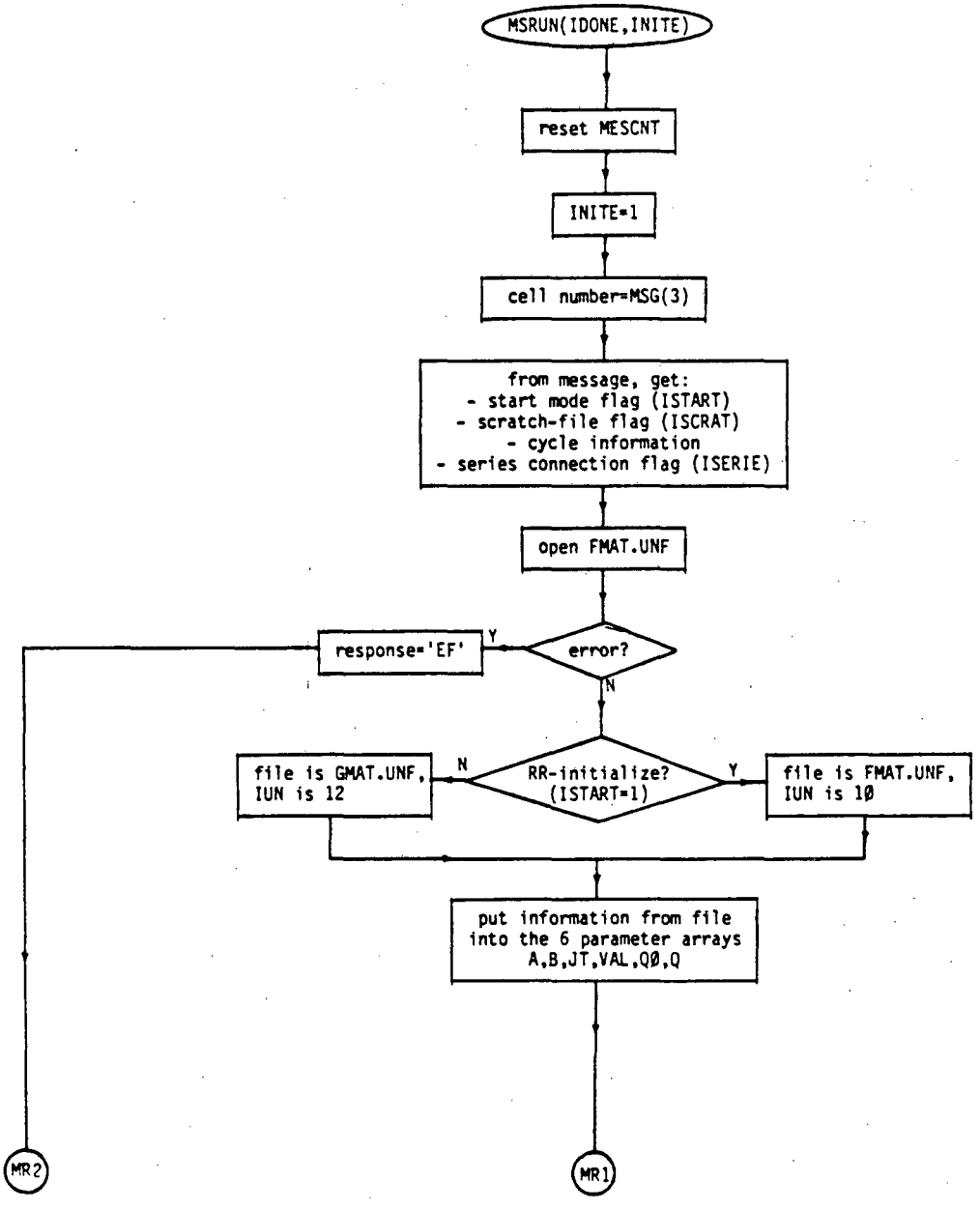

RESET
JMID=24 hrs
JTIP=23:59:50:0
all action timers
action timer=
action timer-JMID
J=1,JMAX
JLAST=JLAST-JMID
JREM=JREM-JMID
JMIN=JMIN-JMID
get time of day
N
time ≤ JTIP?
Y
RETURN

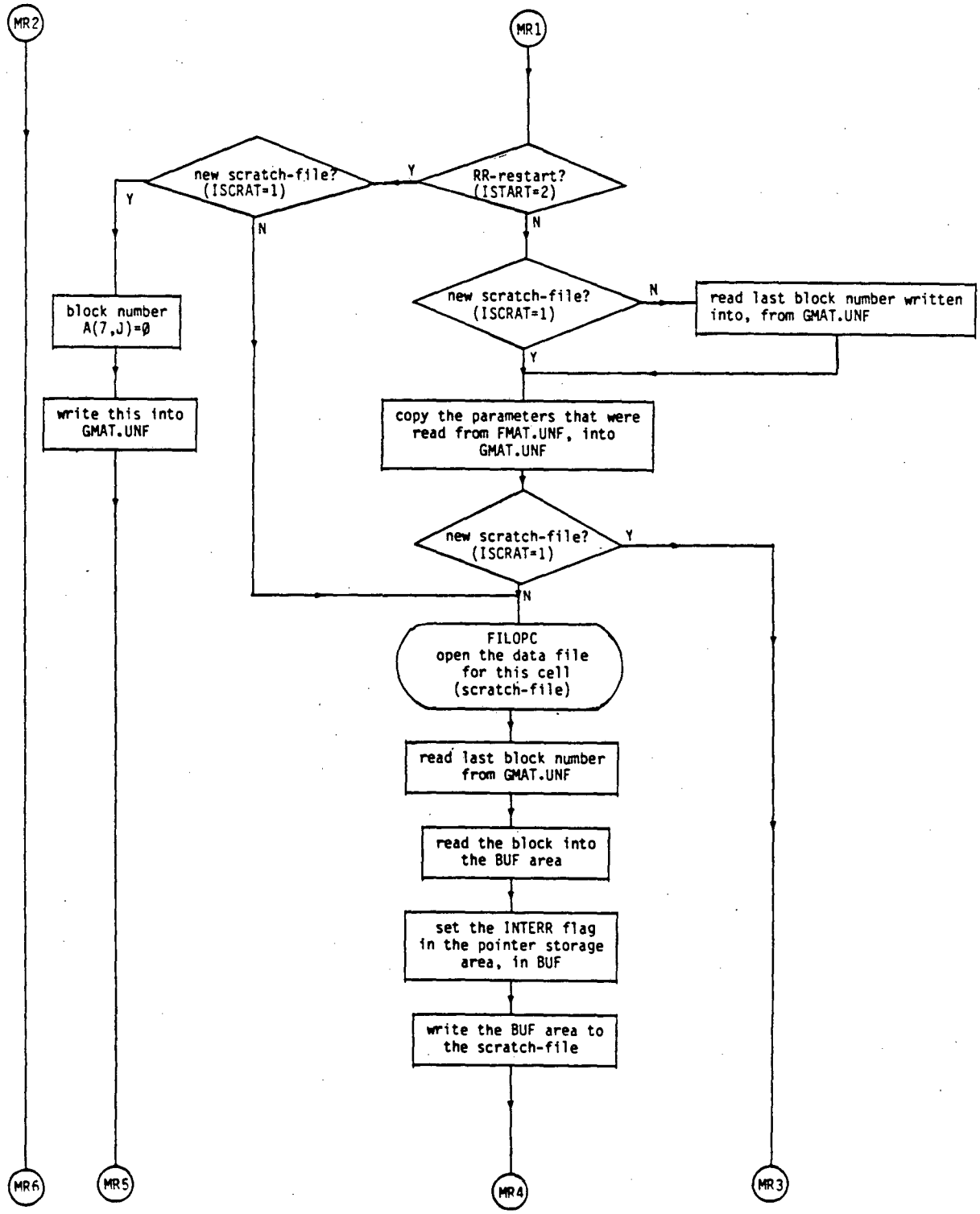

CRMESS
announce that a message is pending
MESCNT=MESCNT+1
RETURN

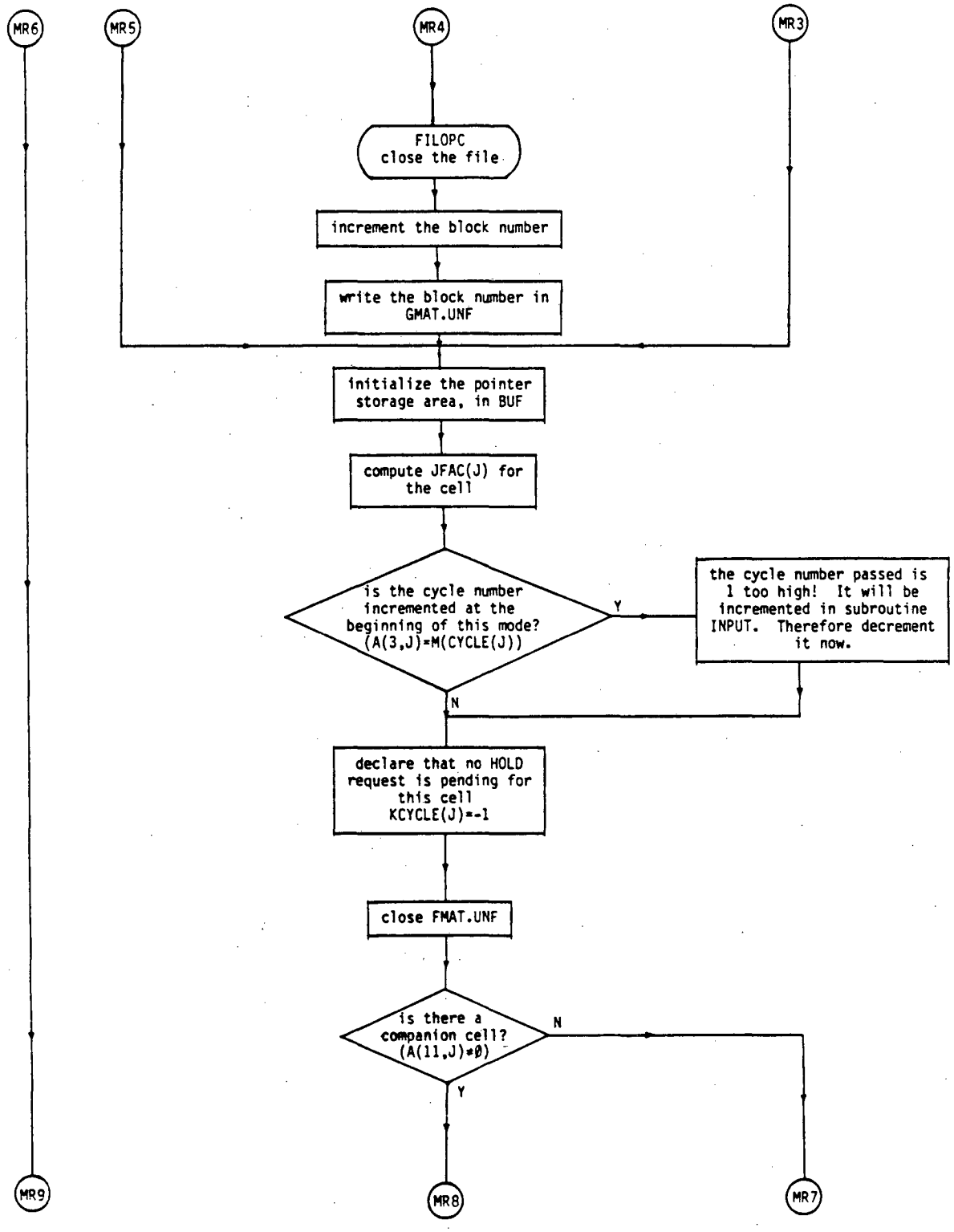

MSBOSS(INITE)
IDONE=Ø
did some cell just go in open-circuit? (IHOLD=1)
Y
MSHOLD(IDONE,MODE=1)
N
did a STOP request expire? (IMSTOP≠Ø)
Y
MSNOW
N
message pending? (MESCNT≠Ø)
N
RETURN
Y
message ID is 'CH'?
Y
MSCHNG(IDONE)
N
'DS'?
Y
MSCHNG(IDONE)
N
'T1'?
Y
MSUPDT(IDONE)
N
'ST'?
Y
MSWHEN(IDONE)
N
'HD'?
Y
MSWHEN(IDONE)
N
MB1

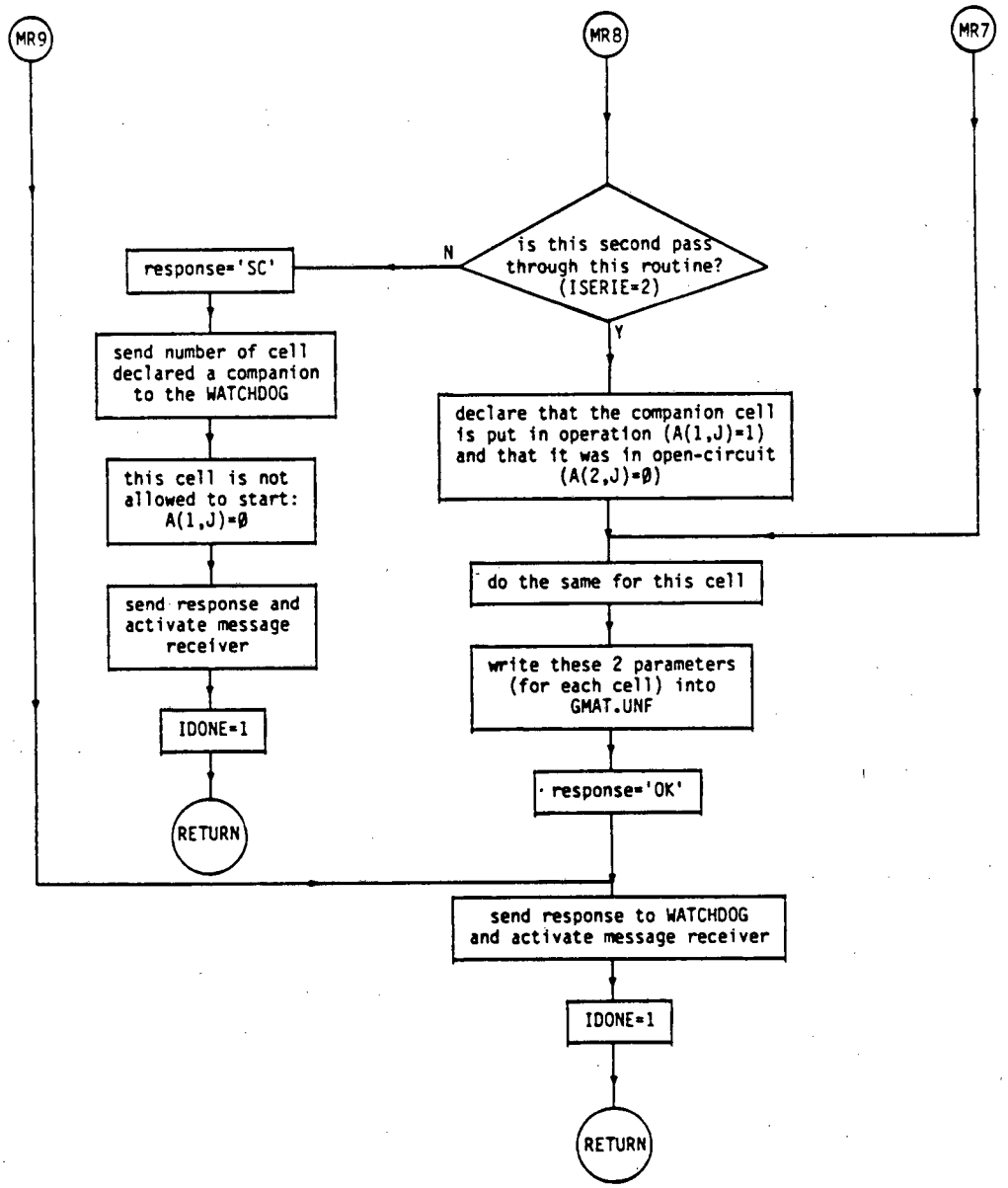

MB1
'DS'?
Y
MSCHNG(IDONE)
N
'T1'?
Y
MSUPDT(IDONE)
N
'ST'?
Y
MSWHEN(IDONE)
N
'HD'?
Y
MSWHEN(IDONE)
N
'T2'?
Y
MSSTOP(IDONE)
N
'T3'?
Y
MSHOLD(IDONE,MODE=Ø)
N
'RR'?
Y
MSWHO(IDONE)
N
'T4'?
Y
MSRUN(IDONE,INITE)
N
did any of them routines execute? (IDONE=1)
N
response is 'BD'
Y
RETURN
send that response back to the Watchdog program
reset MESCNT
activate message receiver
RETURN

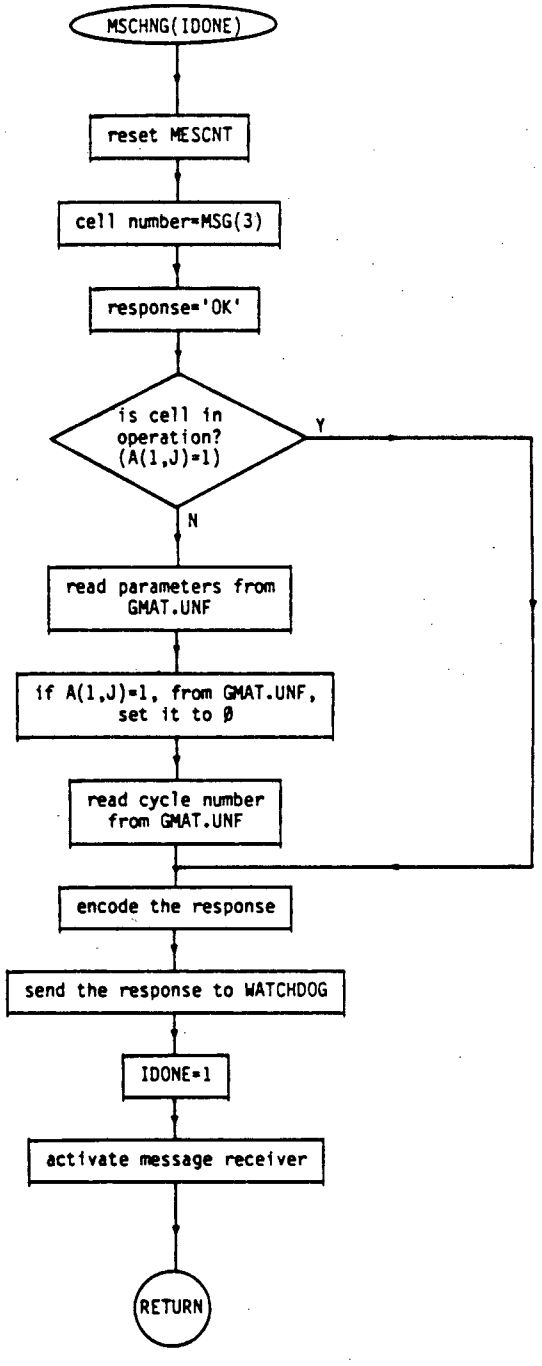

MSWHO(IDONE)
reset MESCNT
cell number=MSG(3)
cell number ≤ JMAX?
N
response='TH'
Y
cell in operation? (A(1,J)≠0)
Y
response='NG'
N
response='OK'
put time and date of interruption in response
put A(1,J) and A(2,J) flags in response
was this information inconsistent?
Y
N
put A(3,J) flag in response
put block number and cycle information in response
send response to WATCHDOG
IDONE=1
activate message receiver
RETURN

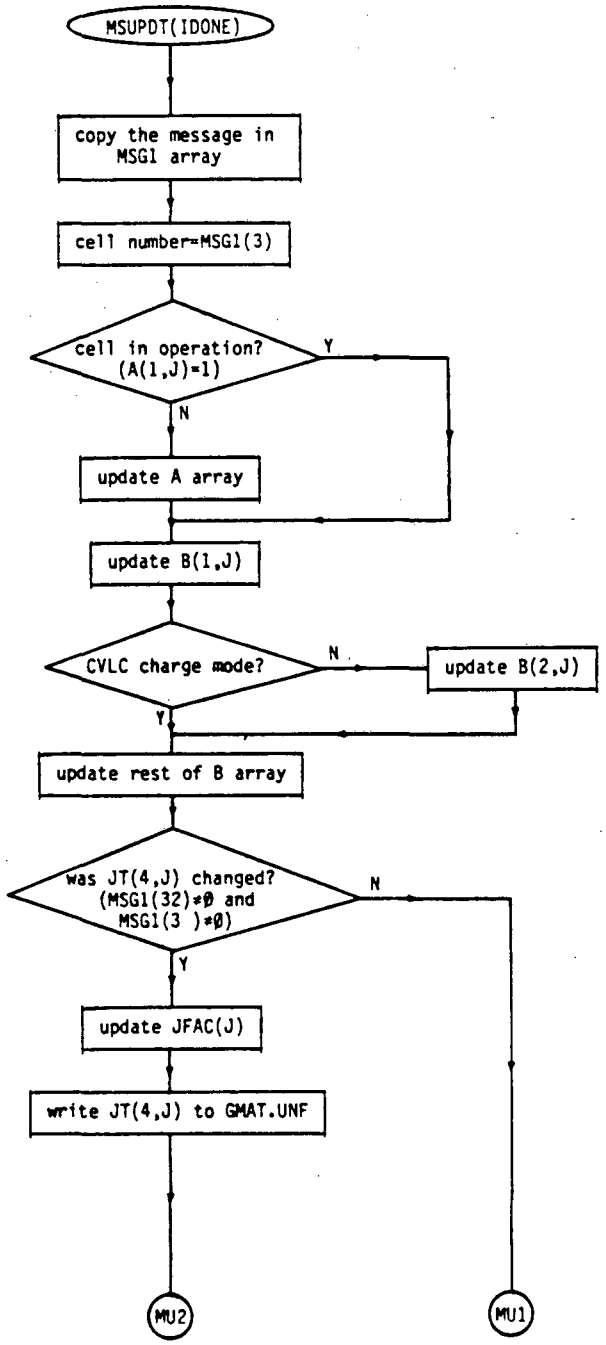

MSRUN(IDONE,INITE)
reset MESCNT
INITE=1
cell number=MSG(3)
from message, get:
- start mode flag (ISTART)
- scratch-file flag (ISCRAT)
- cycle information
- series connection flag (ISERIE)
open FMAT.UNF
error?
Y
response='EF'
N
RR-initialize?
(ISTART=1)
N
file is GMAT.UNF,
IUN is 12
Y
file is FMAT.UNF,
IUN is 10
put information from file
into the 6 parameter arrays
A,B,JT,VAL,QØ,Q
MR2
MR1

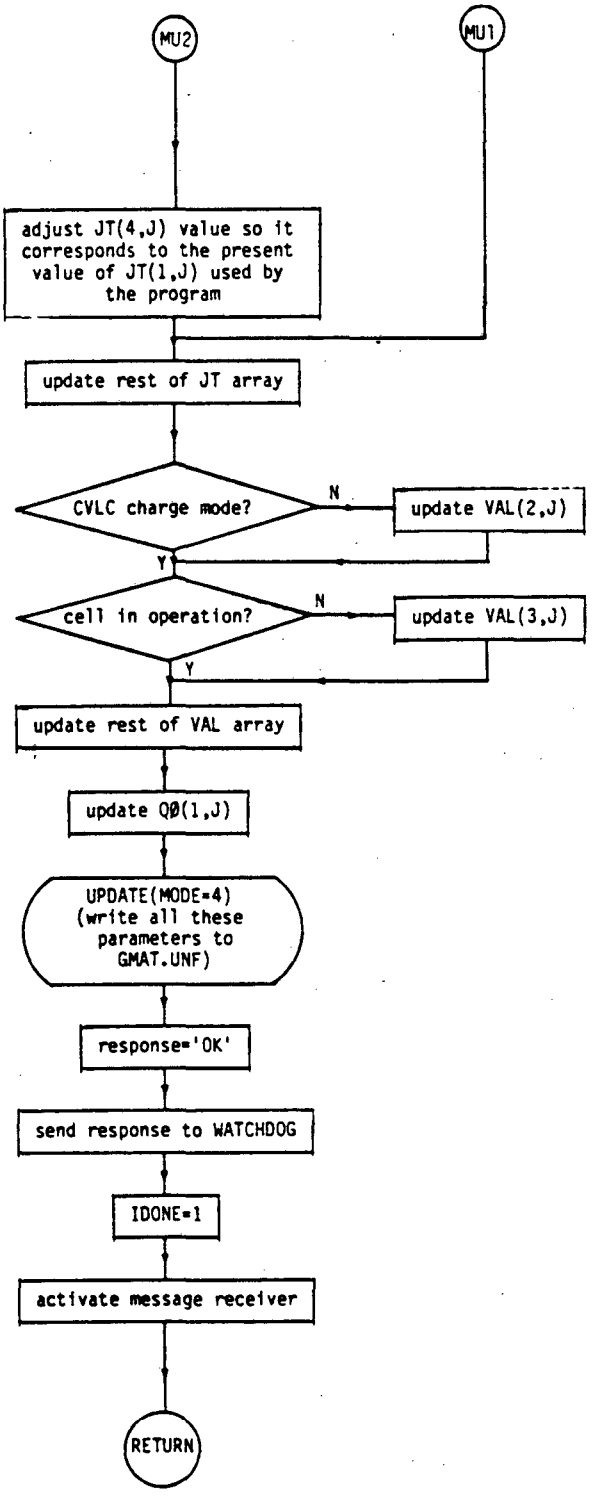

MR2
MR1
new scratch-file?
(ISCRAT=1)
Y
RR-restart?
(ISTART=2)
Y
N
N
block number
A(7,J)=0
new scratch-file?
(ISCRAT=1)
N
read last block number written into, from GMAT.UNF
Y
write this into
GMAT.UNF
copy the parameters that were read from FMAT.UNF, into GMAT.UNF
new scratch-file?
(ISCRAT=1)
Y
N
FILOPC
open the data file
for this cell
(scratch-file)
read last block number from GMAT.UNF
read the block into the BUF area
set the INTERR flag in the pointer storage area, in BUF
write the BUF area to the scratch-file
MR6
MR5
MR4
MR3

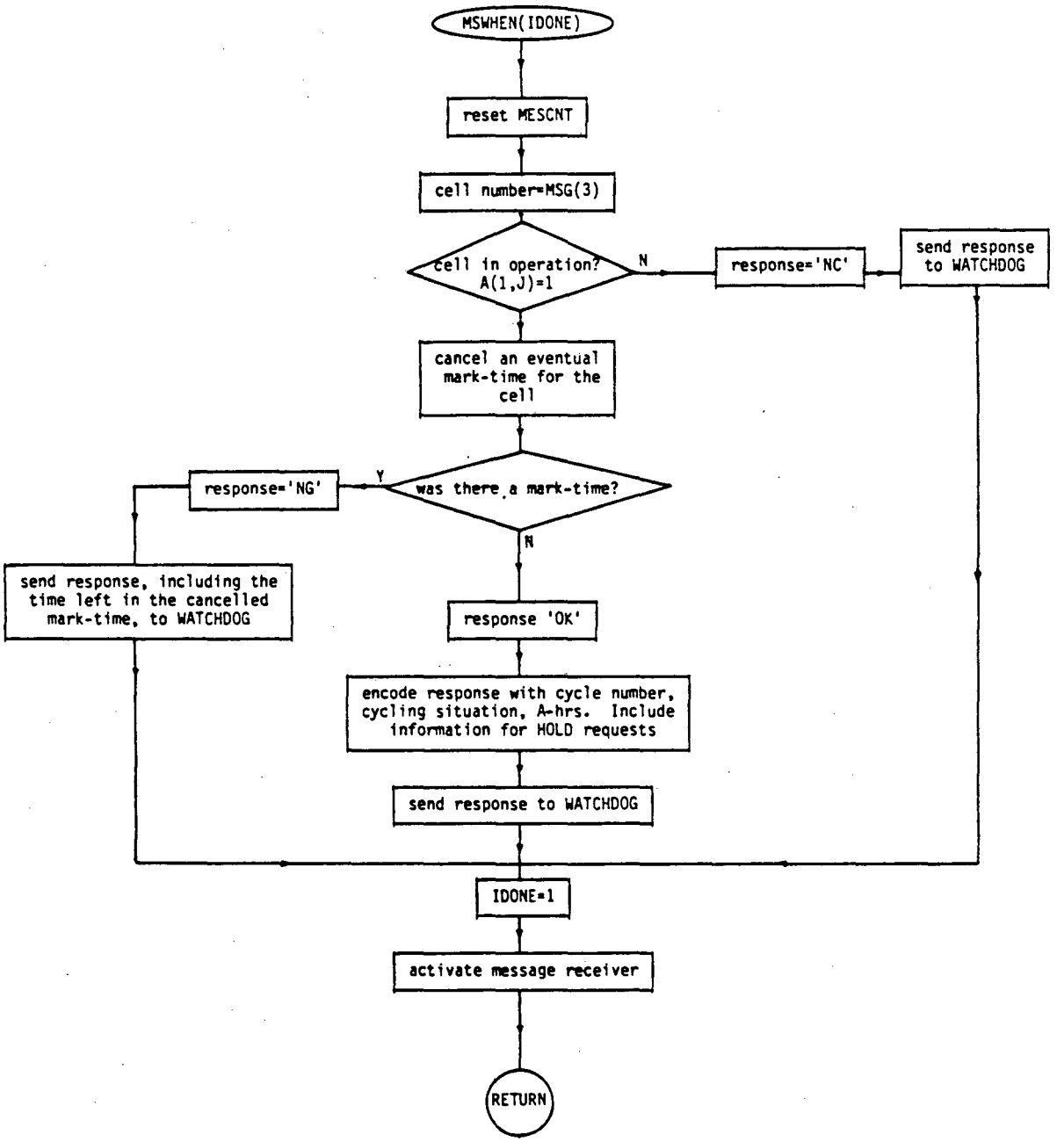

MR6
MR5
MR4
MR3
FILOPC
close the file
increment the block number
write the block number in GMAT.UNF
initialize the pointer storage area, in BUF
compute JFAC(J) for the cell
is the cycle number incremented at the beginning of this mode? (A(3,J)=M(CYCLE(J))
Y
the cycle number passed is 1 too high! It will be incremented in subroutine INPUT. Therefore decrement it now.
N
declare that no HOLD request is pending for this cell KCYCLE(J)=-1
close FMAT.UNF
is there a companion cell? (A(11,J)≠0)
N
Y
MR9
MR8
MR7

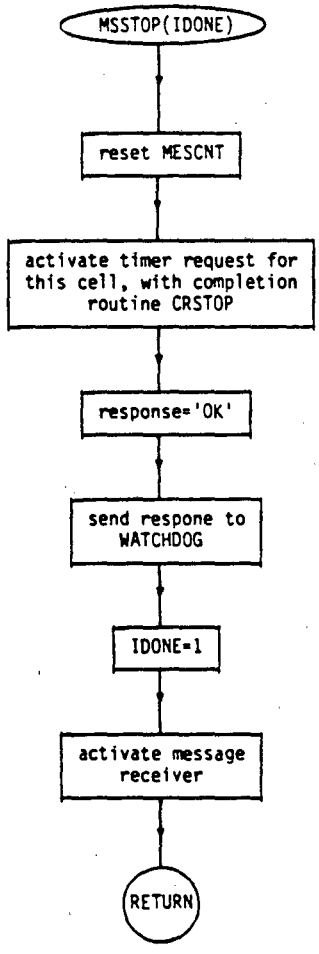

MR9
MR8
MR7
is this second pass through this routine? (ISERIE=2)
N
Y
response='SC'
send number of cell declared a companion to the WATCHDOG
this cell is not allowed to start: A(1,J)=0
send response and activate message receiver
IDONE=1
RETURN
declare that the companion cell is put in operation (A(1,J)=1) and that it was in open-circuit (A(2,J)=0)
do the same for this cell
write these 2 parameters (for each cell) into GMAT.UNF
response='OK'
send response to WATCHDOG and activate message receiver
IDONE=1
RETURN

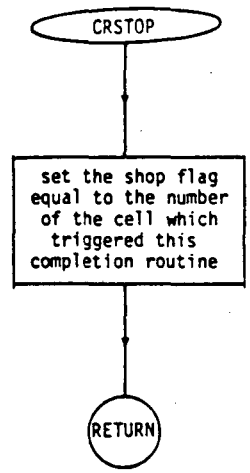

MSCHNG(IDONE)
reset MESCNT
cell number=MSG(3)
response='OK'
is cell in operation? (A(1,J)=1)
Y
N
read parameters from GMAT.UNF
if A(1,J)=1, from GMAT.UNF, set it to ∅
read cycle number from GMAT.UNF
encode the response
send the response to WATCHDOG
IDONE=1
activate message receiver
RETURN

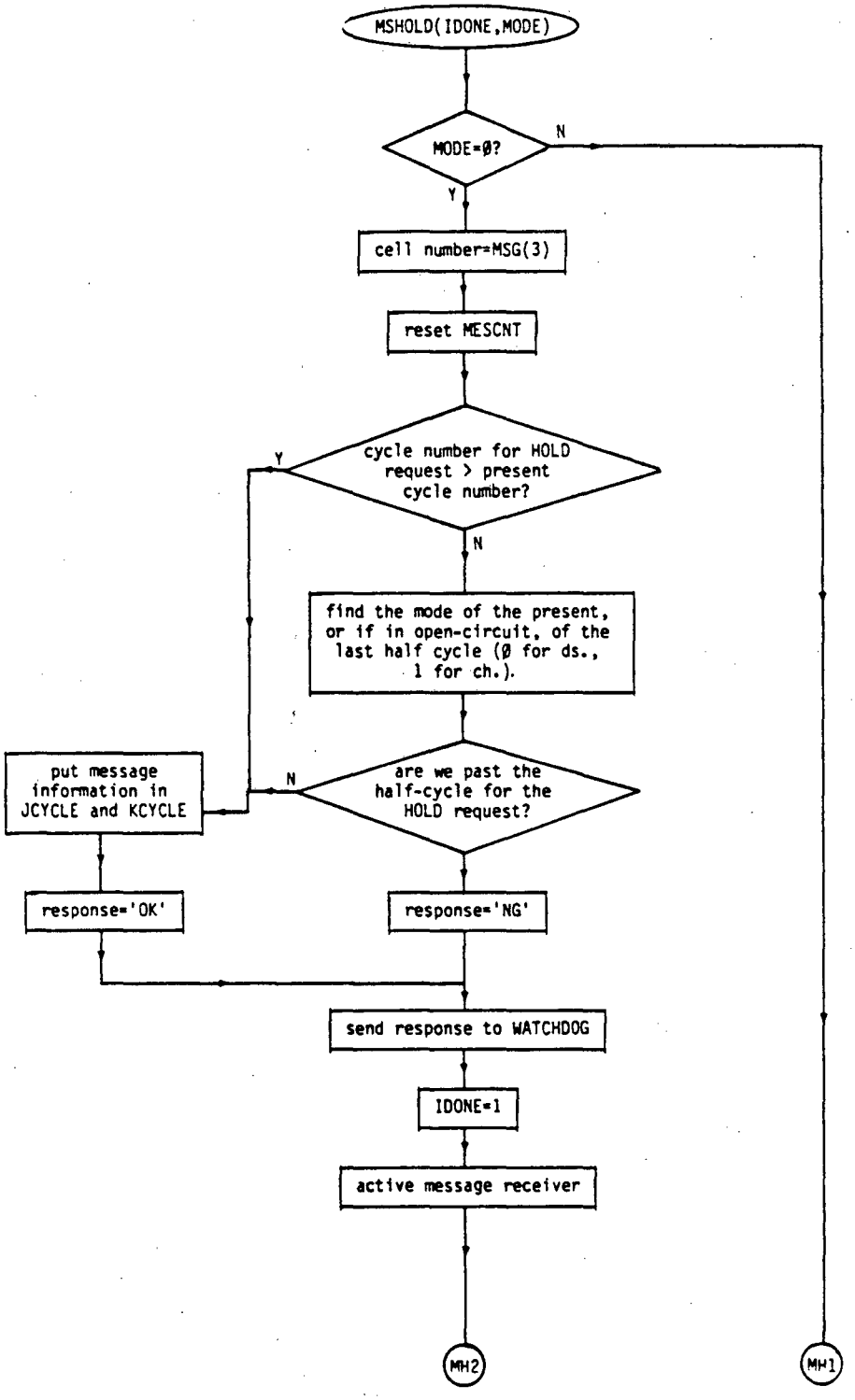

MSUPDT(IDONE)
copy the message in MSG1 array
cell number=MSG1(3)
cell in operation? (A(1,J)=1)
Y
N
update A array
update B(1,J)
CVLC charge mode?
N
update B(2,J)
Y
update rest of B array
was JT(4,J) changed? (MSG1(32)≠Ø and MSG1(3)≠Ø)
N
Y
update JFAC(J)
write JT(4,J) to GMAT.UNF
MU2
MU1

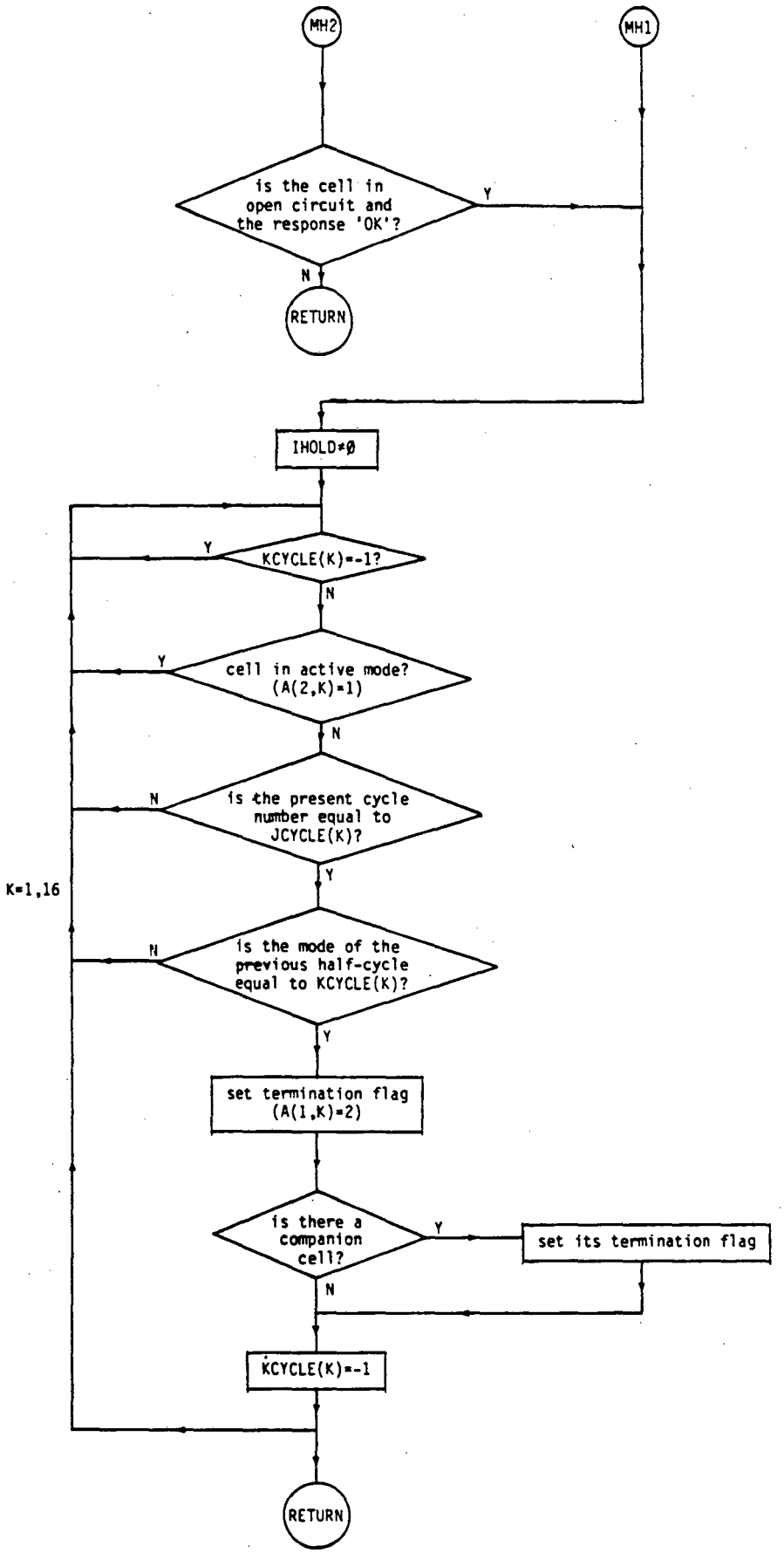

MU2
MU1
adjust JT(4,J) value so it corresponds to the present value of JT(1,J) used by the program
update rest of JT array
CVLC charge mode?
N
update VAL(2,J)
Y
cell in operation?
N
update VAL(3,J)
Y
update rest of VAL array
update QØ(1,J)
UPDATE(MODE=4) (write all these parameters to GMAT.UNF)
response='OK'
send response to WATCHDOG
IDONE=1
activate message receiver
RETURN

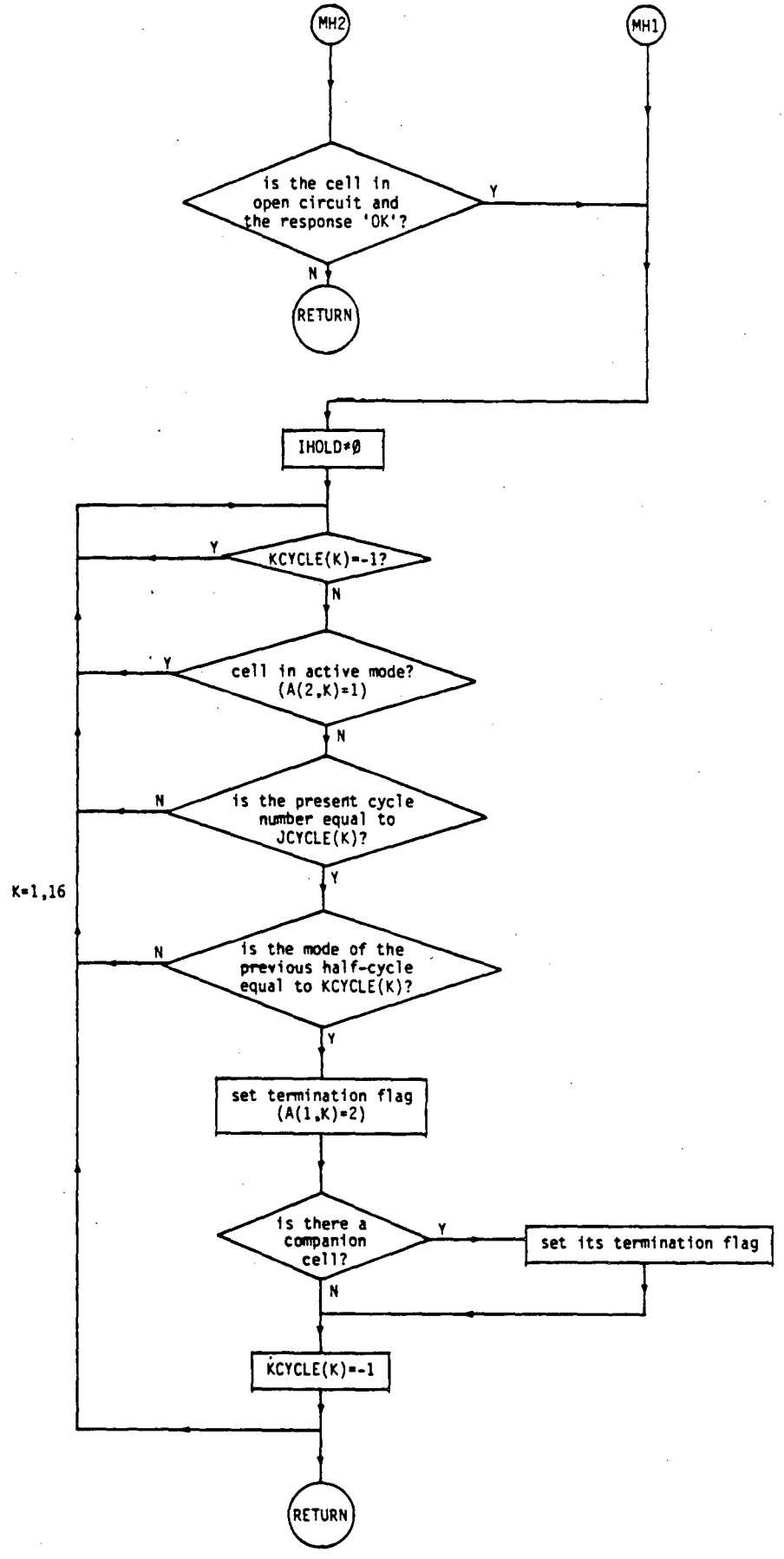

MSWHEN(IDONE)
reset MESCNT
cell number=MSG(3)
cell in operation? A(1,J)=1
N
response='NC'
send response to WATCHDOG
cancel an eventual mark-time for the cell
was there a mark-time?
Y
response='NG'
N
send response, including the time left in the cancelled mark-time, to WATCHDOG
response 'OK'
encode response with cycle number, cycling situation, A-hrs. Include information for HOLD requests
send response to WATCHDOG
IDONE=1
activate message receiver
RETURN

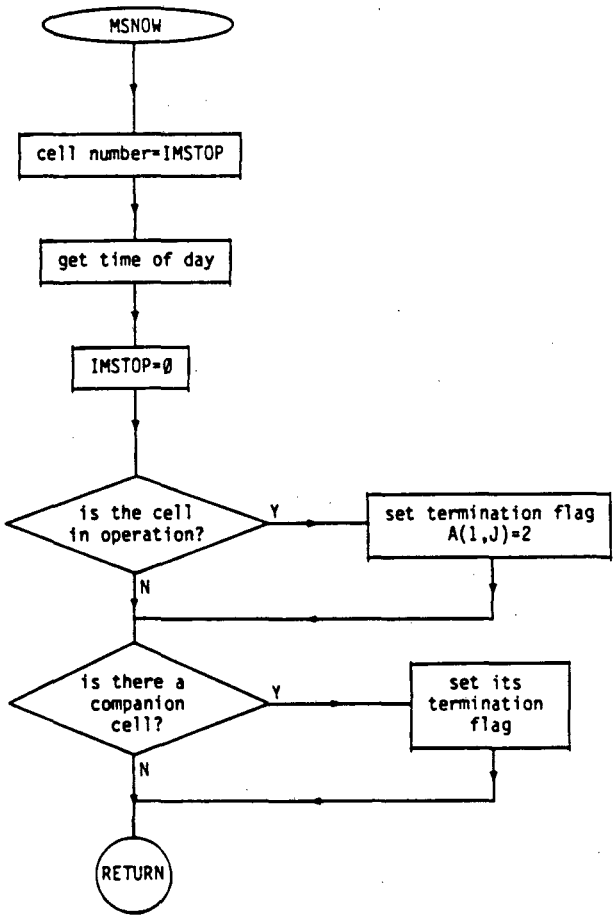

MSSTOP(IDONE)
reset MESCNT
activate timer request for this cell, with completion routine CRSTOP
response='OK'
send respone to WATCHDOG
IDONE=1
activate message receiver
RETURN


CRSTOP
set the shop flag equal to the number of the cell which triggered this completion routine
RETURN


MSHOLD(IDONE,MODE)
MODE=Ø?
N
Y
cell number=MSG(3)
reset MESCNT
cycle number for HOLD request > present cycle number?
Y
N
find the mode of the present, or if in open-circuit, of the last half cycle (Ø for ds., 1 for ch.).
are we past the half-cycle for the HOLD request?
N
put message information in JCYCLE and KCYCLE
response='OK'
response='NG'
send response to WATCHDOG
IDONE=1
active message receiver
MH2
MH1


MH2
MH1
is the cell in open circuit and the response 'OK'?
Y
N
RETURN
IHOLD=0
K=1,16
Y
KCYCLE(K)=-1?
N
Y
cell in active mode? (A(2,K)=1)
N
N
is the present cycle number equal to JCYCLE(K)?
Y
N
is the mode of the previous half-cycle equal to KCYCLE(K)?
Y
set termination flag (A(1,K)=2)
is there a companion cell?
Y
set its termination flag
N
KCYCLE(K)=-1
RETURN


MH2
MH1
is the cell in open circuit and the response 'OK'?
Y
N
RETURN
IHOLD=Ø
K=1,16
KCYCLE(K)=-1?
Y
N
cell in active mode? (A(2,K)=1)
Y
N
is the present cycle number equal to JCYCLE(K)?
N
Y
is the mode of the previous half-cycle equal to KCYCLE(K)?
N
Y
set termination flag (A(1,K)=2)
is there a companion cell?
Y
set its termination flag
N
KCYCLE(K)=-1
RETURN


MSNOW
cell number=IMSTOP
get time of day
IMSTOP=0
is the cell in operation?
Y
set termination flag A(1,J)=2
N
is there a companion cell?
Y
set its termination flag
N
RETURN

```
      PROGRAM START
C     THIS PROGRAM CALLS THE MAIN PROGRAM, IN VIRTUAL MEMORY
      COMMON/HNDLER/MCHAN
      COMMON/CHANNELS/ICHAN(16)
      COMMON/QUEUE/AREAQ
      COMMON/CHACHA/AREAC
      INTEGER AREAQ(100),AREAC(160)
      CALL REALTIM
      END


      SUBROUTINE REALTIM
C     THIS SUBROUTINE IS THE ACTUAL START OF THE PROGRAM
      COMMON/PASS/A,B,JT,VAL,QO,Q
      COMMON/CURTIM/JLAST,JREM
      COMMON/CURCUR/CUR,COMP
      COMMON/PNTRS/KPOINT,LPOINT
      COMMON/RES/JTOP,JMAX
      COMMON/INT4/JTWO,JFOUR,JFAC
      COMMON/TIMTIM/JCHECK,JSET,JCVLC
      COMMON/CHANNELS/ICHAN(16)
      COMMON/CYCLCT/ICYCLE(16),MCYCLE(16)
      COMMON/CCHOLD/JCYCLE(16),KCYCLE(16),IHOLD
      COMMON/HNDLER/MCHAN
      COMMON/QUEUE/AREAQ
      COMMON/CHACHA/AREAC
```

```
      INTEGER A(22,16),B(14,16),DBLK(4),AREAQ(100),AREAC(160)
      INTEGER*4 JT(10,16),JLAST(16),JREM(16),JTOP,JTWO,JFOUR,
     1JCHECK,JSET,JCVLC,JFAC(16)
      DIMENSION VAL(14,16),BUF(128,16),QO(2,16),Q(2,16),CUR(16),
     1COMP(16)
      DATA DBLK/3RHD ,0,0,0/
C     get channel, and do LOOKUP for RT 0 driver
      MCHAN=IGETC()
      IF(MCHAN.LT.0) STOP 'CANNOT ALLOCATE CHANNEL'
      II=LOOKUP(MCHAN,DBLK)
      IF(II.LT.0) STOP 'BAD LOOKUP'
C     call routine that will open the parameter files
      CALL GMOPEN
C     make RT-11 I/O queue larger
      II=IQSET(10,AREAQ)
```

```
C       get 32 I/O channels
        II=ICDFN(32,AREAC)
        IF(II.NE.0) STOP 'NOT ENOUGH FREE SPACE FOR CHANNELS'
C       channel numbers are 16 for cell 1, 17 for cell 2, ...
        DO 1 KK=1,16
1       ICHAN(KK)=15+KK
C       initialize INT*4 parameters
        CALL JTIME(0,1,0,0,JCHECK)
        CALL JJCVT(JCHECK)
        CALL JTIME(0,1,0,0,JSET)
        CALL JJCVT(JSET)
        CALL JTIME(0,0,15,0,JCVLC)
        CALL JJCVT(JCVLC)
        LL=JAFIX(2.,JTWO)
        MM=JAFIX(4.,JFOUR)
C       declare that there are no cells in operation
        DO 5 J=1,JMAX
5       A(1,J)=0
C       go bears !!!
        CALL GO(BUF)
        RETURN
        END



        SUBROUTINE GMOPEN
C       THIS SUBROUTINE OPENS THE PARAMETER FILES
        COMMON/RES/JTOP,JMAX
```

```
      COMMON/PNTRS/KPOINT,LPOINT
      INTEGER*4 JTOP
C     open FMAT.UNF and GMAT.UNF
      OPEN(UNIT=10,NAME='VOL:FMAT.UNF',TYPE='OLD',ACCESS='DIRECT',
     1RECORDSIZE=1,ERR=888,ASSOCIATEVARIABLE=KPOINT)
C     read: maximum number of cells in the test loop
      READ(10'1) JMAX,IDUM
C     close FMAT.UNF
      CLOSE(UNIT=10)
      GOTO 900
888   TYPE *,'ERROR IN OPENING FMAT.UNF'
      STOP
900   OPEN(UNIT=12,NAME='VOL:GMAT.UNF',TYPE='OLD',ACCESS='DIRECT',
     1RECORDSIZE=1,ERR=999,ASSOCIATEVARIABLE=LPOINT)
      RETURN
999   TYPE *,'ERROR IN OPENING GMAT.UNF'
      STOP
      END
```

```
      SUBROUTINE GO(BUF)
C     THIS SUBROUTINE GENERATES THE REAL-TIME LOOP, KEEPING TRACK OF
C     THE TIME AND COMPARING IT TO THE ADJUSTED VALUE OF ACTION TIMERS.
      COMMON/PASS/A,B,JT,VAL,QO,Q
      COMMON/RES/JTOP,JMAX
      COMMON/GDNIT/JMIN
      COMMON/MESSAG/MSG,MESCNT,AREA,AREA2,IMSTOP
      COMMON/CCHOLD/JCYCLE(16),KCYCLE(16),IHOLD
      EXTERNAL RESET,CRMESS
      INTEGER*2 LINK(4)
      INTEGER A(22,16),B(14,16),MSG(59),AREA(4),AREA2(4,16)
      INTEGER*4 JT(10,16),JTOP,JMIN,JSLEEP,J57
      DIMENSION VAL(14,16),QO(2,16),Q(2,16)
      DIMENSION BUF(128,16)
C     activate message receiver
      II=IRCVDF(MSG,3,AREA,CRMESS)
      MESCNT=0
      IHOLD=0
      IMSTOP=0
      ICOUNT=0
C     define smallest action timer = 25 hrs
      CALL JTIME(25,0,0,0,JMIN)
      CALL JJCVT(JMIN)
C     initialize a INT*4 number = 57 ticks
      CALL JTIME(0,0,0,57,J57)
      CALL JJCVT(J57)
      ISTOP=0
```

```
      IFLAG=0
C     get ready to intercept ^C^C's
      CALL SCCA(IFLAG)
C     get time of day
      CALL GTIM(JTOP)
      CALL JJCVT(JTOP)
C
C           hardware check and parameter update loop
C
1     DO 2 K=1,JMAX
      IND=K
C     if cell not in operation, jump
      IF(A(1,K).EQ.0)GOTO 2
C     if not yet time for action, jump
      IF(JCMP(JT(9,K),JTOP).GT.0) GOTO 2
C     check the hardware
      CALL CHECK(IND)
```

```
C       update the parameters in GMAT.UNF
        CALL UPDATE(IND,3)
C       if no error flag set, jump
        IF(A(1,K).NE.2) GOTO 2
C       take data, no limits
        CALL INPUT(IND,IEX,2,BUF)
C       terminate the operation for the cell
        CALL FINISH(IND,0,BUF)
2       CONTINUE
C
C               status change loop
C
3       DO 5 K=1,JMAX
        IND=K
        ICOUNT=ICOUNT+1
C       if cell not in operation, jump
        IF(A(1,K).EQ.0) GOTO 5
C       if not yet time for action, jump
        IF(JCMP(JT(3,K),JTOP).GT.0) GOTO 5
C       status change, mode 1
        CALL CHANGE(IND,1,IOUT,BUF)
C       terminate control mode
        IF(IOUT.EQ.1) CALL CHOOSE(IND,3,ISTOP,BUF)
C       take data, no limits
        IF(IOUT.EQ.1) CALL INPUT(IND,IEX,2,BUF)
C       status change, mode 2
        IF(IOUT.EQ.1) CALL CHANGE(IND,2,IOUT,BUF)
        IF(IOUT.EQ.2) GOTO 5
        IF(IOUT.NE.3) GOTO 4
        ISTOP=0
```

```
C       initialize control mode
        CALL CHOOSE(IND,1,ISTOP,BUF)
C       if no error, jump
        IF(ISTOP.EQ.0) GOTO 4
C       terminate operation
        A(2,K)=0
        CALL FINISH(IND,0,BUF)
        ISTOP=0
        GOTO 5
C       status change, mode 3
4       IF(IOUT.EQ.3) CALL CHANGE(IND,3,IOUT,BUF)
C       write BUF to file
        IF(IOUT.EQ.4) CALL INPUT(IND,IEX,1,BUF)
C       status change, mode 4
        IF(IOUT.EQ.4) CALL CHANGE(IND,4,IOUT,BUF)
C       store date and cycle number
        IF(IOUT.EQ.5) CALL INPUT(IND,IEX,3,BUF)
```

```
C       update parameters in GMAT.UNF
        CALL UPDATE(IND,1)
5       CONTINUE
C       make a second pass for all cells
        IF(ICOUNT.EQ.JMAX) GOTO 3
        ICOUNT=0
C
C               current control loop
C
        DO 7 K=1,JMAX
        IND=K
C       if cell not in operation, jump
        IF(A(1,K).EQ.0) GOTO 7
C       if not yet time for action, jump
        IF(JCMP(JT(2,K),JTOP).GT.0) GOTO 7
C       do current control
        CALL CHOOSE(IND,2,ISTOP,BUF)
C       if no error, jump
        IF(ISTOP.EQ.0) GOTO 7
        ISTOP=0
C       get ready to change cell status
        II=JMOV(JTOP,JT(3,K))
        GOTO 3
7       CONTINUE
C
C               data-taking loop
C
        DO 10 K=1,JMAX
        IND=K
C       if cell not in operation, jump
```

```
      IF(A(1,K).EQ.0) GOTO 10
C     if not yet time for action, jump
      IF(JCMP(JT(1,K),JTOP).GT.0) GOTO 10
C     take data, check limits
      CALL INPUT(IND,IEX,0,BUF)
C     if no limit exceeded, jump
      IF(IEX.NE.1) GOTO 10
      IEX=0
C     get ready to change cell status
      II=JMOV(JTOP,JT(3,K))
      GOTO 3
10    CONTINUE
C     find smallest action timer
      CALL GDNITE(0,INITE,J57)
C     if no ^C^C has been detected, terminate
15    IF(IFLAG.EQ.0) GOTO 25
      DO 20 K=1,JMAX
      IND=K
```

```
C       if cell not in operation, jump
        IF(A(1,K).EQ.0) GOTO 20
C       take data, no limits
        CALL INPUT(IND,IEX,2,BUF)
20      CONTINUE
C       terminate operation for all cells
        CALL FINISH(JMAX,1,BUF)
25      CONTINUE
C       give control to message handler
        CALL MSBOSS(INITE,BUF)
C       if no sleep, jump to the beginning of the real-time loop
        IF(INITE.EQ.1) GOTO 1
        DO 30 K=1,JMAX
        IND=K
C       if terminatin flag not set, jump
        IF(A(1,K).NE.2) GOTO 30
C       take data, no limits
        CALL INPUT(IND,IEX,2,BUF)
C       terminate cell operation
        CALL FINISH(IND,0,BUF)
30      CONTINUE
C       go and take a nap, if you can
        CALL GDNITE(1,INITE,J57)
C       if no more sleep, jump back to start of real-time loop
        GOTO(15,1) INITE+1
        RETURN
        END
```

```
      SUBROUTINE GDNITE(IGD,INITE,J57)
C     THIS SUBROUTINE DETERMINES IF A CELL IS ALLOWED TO SLEEP, AND
C     IF SO, FOR HOW LONG
      COMMON/PASS/A,B,JT,VAL,QO,Q
      COMMON/RES/JTOP,JMAX
      COMMON/GDNIT/JMIN
      INTEGER A(22,16),B(14,16)
      INTEGER*4 JT(10,16),JTOP,JMIN,JSLEEP,J57
      DIMENSION VAL(14,16),QO(2,16),Q(2,16)
      IF(IGD.EQ.1) GOTO 20
      DO 15 K=1,JMAX
C     if cell not in operation, jump
      IF(A(1,K).EQ.0) GOTO 15
C     get minimal action timer
      DO 10 L=1,3
10    IF(JCMP(JT(L,K),JMIN).LT.0) II=JMOV(JT(L,K),JMIN)
      IF(JCMP(JT(9,K),JMIN).LT.0) II=JMOV(JT(9,K),JMIN)
```

```
15      CONTINUE
        INITE=0
        TYPE 1000
1000    FORMAT(/)
        RETURN
C       get time of day and convert
20      CALL GTIM(JTOP)
        CALL CVTTIM(JTOP,IHR,IMIN,ISCND,ITICK)
        CALL JJCVT(JTOP)
C       if time is 23:59:55, call RESET
        IF(IHR.EQ.23.AND.IMIN.EQ.59.AND.ISCND.GT.55) CALL RESET(0)
C       define sleep time
        II=JSUB(JMIN,JTOP,JSLEEP)
C       if no sleep, jump
        IF(II.LE.0) GOTO 25
        JJ=JCMP(JSLEEP,J57)
        CALL JJCVT(JSLEEP)
C       if sleep time > 57 ticks, sleep for that time. Otherwise,
C       sleep for less time.
        IF(JJ.GT.0) CALL ISLEEP(0,0,0,57)
        IF(JJ.LE.0) CALL ITWAIT(JSLEEP)
        INITE=0
        TYPE 2000,IHR,IMIN,ISCND
2000    FORMAT('+',T2,I2,':',I2,':',I2)
        RETURN
C       define minimal action timer = 25 hrs
25      CALL JTIME(25,0,0,0,JMIN)
        CALL JJCVT(JMIN)
        INITE=1
        RETURN
```

```
      END



      SUBROUTINE ANNOUNC(B)
C     THIS SUBROUTINE GIVES THE TIME IN HRS,MINS,SECS AND TICKS
      INTEGER*4 B
      CALL JJCVT(B)
      CALL CVTTIM(B,IHR,IMIN,ISEC,ITICK)
      TYPE 9000,IHR,IMIN,ISEC,ITICK
9000  FORMAT('0',3X,I2,':',I2,':',I2,':',I2)
      RETURN
      END
```

```
        .TITLE IDATE
        .GLOBL IDATE
;+
;THIS PROGRAM CAN BE ASSEMBLED SEPARATELY AND LINKED WITH A USER PROGRAM.
;IT MUST BE CALLED FROM A FORTRAN PROGRAM
;       CALL IDATE(IMONTH,IDAY,IYEAR)
;
;INPUT: NONE
;
;OUTPUT: IMONTH (1-12)
;        IDAY (1-31)
;        IYEAR
;
;ERRORS: R0=0 IF NO DATE ENTERED
;REF: RT-11 V0.4 MANUAL 3A, 2-26
;-
        .MCALL  .DATE
IDATE:  .DATE
        MOV     R0,R2
        BEQ     1$
        BIC     #177740,R2
        ADD     #72.,R2
        MOV     R0,R1
        ASL     R1
        ASL     R1
        ASL     R1
        SWAB    R1
        BIC     #177740,R1
```

```
        SWAB    R0
        ASR     R0
        ASR     R0
        BIC     #177740,R0
        MOV     R0,@2(R5)
        MOV     R1,@4(R5)
        MOV     R2,@6(R5)
1$:     RETURN

        .END    IDATE
```

```
        SUBROUTINE CHANGE(J,IIN,IOUT,BUF)
C       THIS SUBROUTINE IMPLEMENTS THE CHANGE OF OPERATING MODE OF
C       A CELL DURING THE CYCLING SEQUENCE. IT REDEFINES A TIMER FOR
```

```
C       COMPLETION OF THE NEWLY ENTERED MODE.
        COMMON/PASS/A,B,JT,VAL,QO,Q
        COMMON/TIMTIM/JCHECK,JSET,JCVLC
        COMMON/RES/JTOP,JMAX
        COMMON/OFFSET/OFF(16)
        COMMON/CCHOLD/JCYCLE(16),KCYCLE(16),IHOLD
        INTEGER A(22,16),B(14,16)
        INTEGER*4 JT(10,16),JTOP,BB,JCHECK,JSET,JCVLC
        DIMENSION VAL(14,16),QO(2,16),Q(2,16)
        DIMENSION BUF(128,16)
D       TYPE *,'CHANGE'
        CALL GTIM(JTOP)
        CALL JJCVT(JTOP)
        GOTO(5,35,40,50) IIN
C       start of a new mode
5       A(8,J)=0
C       next time for INPUT is now
        II=JMOV(JTOP,JT(1,J))
        II=JMOV(JTOP,BB)
        CALL ANNOUNC(BB)
C       ring bell to announce that something is changing
10      IF(ITTOUR("007).NE.0) GOTO 10
C       if cell was in open circuit, jump
        IF(A(2,J).EQ.0) GOTO 15
        IOUT=1
        RETURN
C       check if controller is in computer mode
15      CALL GET(ICOM,A(9,J))
C       if yes, jump
        IF(ICOM.EQ.0) GOTO 30
```

```
C       if no, wait one more open-circuit period
        II=JADD(JTOP,JT(7,J),JT(3,J))
        II=JMOV(JT(3,J),JT(2,J))
        II=JMOV(JT(2,J),JT(1,J))
C       ring bell to wake the user up
        DO 25 I=1,5
20      IF(ITTOUR("007).NE.0) GOTO 20
25      CONTINUE
        TYPE 1000,A(9,J)
1000    FORMAT(1X,'PLEASE PUT CURRENT CONTROLLER ',I2,' BACK IN ',
       1'COMPUTER MODE')
        IOUT=2
        RETURN
C       we start an active mode
30      A(2,J)=1
        IOUT=3
        RETURN
```

```
C       go in open circuit
35      A(2,J)=0
        IHOLD=1
C       define next status of cell (charge or discharge: opposite to
C       the one it exited from)
        A(3,J)=1-A(3,J)
C       adjust action timer for end of open circuit and for next current
C       control of the cell
        II=JADD(JTOP,JT(7,J),JT(3,J))
        II=JMOV(JT(3,J),JT(2,J))
        II=JMOV(JT(3,J),BB)
        IF(A(3,J).EQ.0) TYPE 1100,J
        IF(A(3,J).EQ.1) TYPE 1200,J
1100    FORMAT(1X,'CELL ',I2,' IS NOW IN OPEN-CIRCUIT. ',
        1'I WILL START A DISCHARGE AT')
1200    FORMAT(1X,'CELL ',I2,' IS NOW IN OPEN-CIRCUIT. ',
        1'I WILL START A CHARGE AT')
        CALL ANNOUNC(BB)
        GOTO 45
C       adjust action timer for end of charge or discharge (will be
C       irrelevant if no time limit was requested (only single modes):
C       the value added will be 24 hrs)
40      II=JADD(JTOP,JT(6-A(3,J),J),JT(3,J))
        II=JMOV(JTOP,JT(2,J))
        IF(A(3,J).EQ.0) TYPE 1300,J
        IF(A(3,J).EQ.1) TYPE 1400,J
1300    FORMAT(1X,'I HAVE STARTED A DISCHARGE OF CELL ',I2)
1400    FORMAT(1X,'I HAVE STARTED A CHARGE OF CELL ',I2)
C       define offset M in the pointer storage area
C       M = 1 for charge, 2 for open circuit, 3 for discharge
```

```
45      M=1+(1-A(2,J))+A(2,J)*(2-2*A(3,J))
C       if the pointer for this mode, in the storage area, is 0, jump
        IF(BUF(120+M,J).EQ.0) GOTO 55
C       we have to start a new block in the scratch-file
C       define the end of the block as the last data-record stored
        BUF(124,J)=-A(6,J)
        IOUT=4
        RETURN
C       define M (see above)
50      M=1+(1-A(2,J))+A(2,J)*(2-2*A(3,J))
C       define pointer to the first data-record for this mode, in the
C       BUF storage area ( = data block)
55      BUF(120+M,J)=A(6,J)+1
        IF(A(2,J).EQ.0) IOUT=6
        IF(A(2,J).EQ.1) IOUT=5
C       if no companion, jump
        IF(A(11,J).EQ.0) GOTO 57
C       if not controlling cell, return
        IF(A(2,A(11,J)).NE.A(2,J)) RETURN
```

```
C       set current to 0
57      CALL PLACE(A(9,J),0.,A(10,J),IDUM)
C       if cell was in pulse mode and goes in open circuit, set
C       the controller in the normal mode
        IF(A(2,J).EQ.0.AND.B(4-A(3,J),J).GE.7) CALL EXT(0,A(9,J))
C       disconnect the controller
        CALL SET(0,0,A(9,J))
C       if cell goes in open circuit, return
        IF(A(2,J).EQ.0) RETURN
C       sleep for 3 ticks and measure current offset
        CALL ISLEEP(0,0,0,3)
        OFF(J)=RLOGG(A(19,J),A(20,J),16,2000)
        TYPE *,'OFFSET ',OFF(J),' V'
C       if there is a companion, update its offset
        IF(A(11,J).NE.0) OFF(A(11,J))=OFF(J)
C       connect the controller
        CALL SET(A(2,J),A(3,J),A(9,J))
        RETURN
        END



        SUBROUTINE CHECK(J)
C       THIS SUBROUTINE CHECKS IF THE HARDWARE IS PROPERLY OPERATING
        COMMON/PASS/A,B,JT,VAL,Q0,Q
        COMMON/TIMTIM/JCHECK,JSET,JCVLC
        COMMON/RES/JTOP,JMAX
        INTEGER*4 JT(10,16),JTOP,JCHECK,JSET,BB,JCVLC
```

```
      INTEGER A(22,16),B(14,16)
      DIMENSION VAL(14,16),Q0(2,16),Q(2,16)
D     TYPE *,'CHECK'
C     get time of day
      CALL GTIM(JTOP)
      CALL JJCVT(JTOP)
C     update action timer
      II=JADD(JTOP,JCHECK,JT(9,J))
C     check if controller in computer mode
      CALL GET(ICOM,A(9,J))
C     if cell in open circuit, it does not matter: return
      IF(A(2,J).EQ.0) RETURN
C     if not in computer mode, jump
      IF(ICOM.EQ.1) GOTO 5
C     if the current has the wrong sign, jump
      CURR=RLOGG(A(19,J),A(20,J),8,0)
      IF(((A(3,J).EQ.0.AND.CURR.LT.0.).OR.(A(3,J).EQ.1.AND.CURR.GT.0.
      1))) GOTO 5
      RETURN
```

```
C       ring bell
5       DO 10 I=1,5
7       IF(ITTOUR("007).NE.0) GOTO 7
10      CONTINUE
        TYPE 1000,J
1000    FORMAT(1X,'CELL ',I2,' HAS BEEN REMOVED FROM CIRCUIT')
C       set termination flag
        A(1,J)=2
        IF(A(11,J).EQ.0) RETURN
C       if there is a companion cell, and it is still in operation,
C       force it to call this routine ASAP
        IF(A(1,A(11,J)).NE.0) II=JMOV(JTOP,JT(9,A(11,J)))
        RETURN
        END



        SUBROUTINE CHOOSE(J,ISEC,ISTOP,BUF)
C       THIS ROUTINE CALLS THE APPROPRIATE CONTROL ROUTINE FOR A CELL
        COMMON/PASS/A,B,JT,VAL,Q0,Q
        COMMON/RES/JTOP,JMAX
        INTEGER A(22,16),B(14,16)
        INTEGER*4 JT(10,16),JTOP
        DIMENSION VAL(14,16),Q0(2,16),Q(2,16)
        DIMENSION BUF(128,16)
D       TYPE 1000
1000    FORMAT(1X,'CHOOSE')
C       if control mode exists, jump
```

```
      IF(B(3+A(3,J),J).GT.0.AND.B(3+A(3,J),J).LE.9) GOTO 2
C     set error flag
      ISTOP=1
      RETURN
2     GOTO(5,50) A(3,J)+1
C
C           discharge control modes
C
5     GOTO(10,10,10,15,15,15,20,20,20) B(3,J)
10    CALL CC(J,ISEC,ISTOP,BUF)
      RETURN
15    CALL CP(J,ISEC,ISTOP,BUF)
      RETURN
20    CALL EXCON(J,ISEC,ISTOP,BUF)
      RETURN
C
C           charge control modes
C
50    GOTO(55,55,55,70,70,70,75,75,75) B(4,J)
55    CALL CC(J,ISEC,ISTOP,BUF)
D     TYPE *,'END OF CHOOSE'
      RETURN
```

```
65      CONTINUE
70      CALL CVLC(J,ISEC,ISTOP,BUF)
        RETURN
75      CALL EXCON(J,ISEC,ISTOP,BUF)
        RETURN
        END



        SUBROUTINE CC(J,ISEC,ISTOP,BUF)
C       THIS SUBROUTINE ORGANIZES THE CONSTANT CURRENT OPERATION
        COMMON/PASS/A,B,JT,VAL,QO,Q
        COMMON/CURTIM/JLAST,JREM
        COMMON/CURCUR/CUR,COMP
        COMMON/CYCLCT/ICYCLE(16),MCYCLE(16)
        COMMON/RES/JTOP,JMAX
        COMMON/OFFSET/OFF(16)
        INTEGER A(22,16),B(14,16)
        INTEGER*4 JT(10,16),JLAST(16),JREM(16),JDIF,JTOP,BB
        DIMENSION VAL(14,16),QO(2,16),Q(2,16),CUR(16),COMP(16)
        DIMENSION BUF(128,16)
        LOGICAL*1 ANS
C       get time of day
        CALL GTIM(JTOP)
        CALL JJCVT(JTOP)
        GOTO(5,10,20) ISEC
C
C               initialization section
```

```
C
C       if single charge or discharge, jump
5       IF(B(3+A(3,J),J).EQ.2) GOTO 8
C       define the A-hrs to pas on charge
        IF(A(3,J).EQ.1) Q0(2,J)=VAL(4,J)*Q(1,J)
8       CONTINUE
D       TYPE 900
900     FORMAT(1X,'CC INIT')
C       if no cells in series, return
        IF(B(A(3,J)+3,J).NE.3) RETURN
C       if cell in series is still in open circuit, force it to start
C       a charge or a discharge
        IF(A(2,A(11,J)).EQ.0) II=JMOV(JTOP,JT(3,A(11,J)))
        RETURN
C
C               control section
C
C       if start of mode, compute current to be set
10      IF(A(8,J).EQ.0) CUR(J)=VAL(11+A(3,J),J)*A(10,J)/5.
D       TYPE 1000
1000    FORMAT(1X,'CC  CONTROL')
```

```
C       update action timer
        II=JADD(JTOP,JT(8,J),JT(2,J))
C       if new mode, jump
        IF(A(8,J).EQ.0) GOTO 14
C       if cell not controlling, jump
        IF(J.LT.A(11,J)) GOTO 13
C       compute A-hrs for this cell, and eventually for its companion
        II=JSUB(JTOP,JLAST(J),JDIF)
        CALL JJCVT(JDIF)
        CALL CVTTIM(JDIF,IHR,IMIN,ISCND,ITICK)
        HR=FLOAT(IHR)+FLOAT(IMIN)/60.+FLOAT(ISCND)/3600.+FLOAT(ITICK)/
        1216000.
        READCH=(RLOGG(A(19,J),A(20,J),8,0)-OFF(J))*VAL(3,J)
        Q(A(3,J)+1,J)=Q(A(3,J)+1,J)+HR*ABS(READCH)
        TYPE *,'CELL ',J,', A-HRS PASSED  ',Q(A(3,J)+1,J)*A(10,J)/5.
        IF(B(3+A(3,J),J).NE.3) GOTO 12
        Q(A(3,A(11,J))+1,A(11,J))=Q(A(3,A(11,J))+1,A(11,J))+HR*ABS(READCH)
        TYPE *,'CELL ',A(11,J),', A-HRS PASSED  ',Q(A(3,A(11,J))+1,
        1A(11,J))*A(10,A(11,J))/5.
C       update JLAST timers
        II=JMOV(JTOP,JLAST(A(11,J)))
12      II=JMOV(JTOP,JLAST(J))
C       if A-hrs exceeded, set ISTOP flag
13      IF(Q(A(3,J)+1,J).LT.Q0(A(3,J)+1,J)) GOTO 15
        ISTOP=1
        RETURN
C       update JLAST timer
14      II=JMOV(JTOP,JLAST(J))
C       if cell not controlling, return
15      IF(J.LT.A(11,J)) RETURN
```

```
C       define setpoint and accuracy for current regulation, call CALIB
        SETPNT=VAL(11+A(3,J),J)*(1-2*A(3,J))/VAL(3,J)
        ACC=.001
        CALL CALIB(SETPNT,CUR(J),A(9,J),A(10,J),A(19,J),A(20,J),OFF(J),
        1ACC,.005,A(8,J),ANS)
C       if ANS = F, set termination flag for this cell, and eventually
C       for its companion
        IF(ANS.EQ.'F') A(1,J)=2
        IF(ANS.EQ.'F'.AND.A(11,J).NE.0) A(1,A(11,J))=2
        RETURN
C
C               termination section
C
20      CONTINUE
D       TYPE 1100
1100    FORMAT(1X,'CC TERMIN')
C       if ISTOP was set in the control section (A-hrs exceeded), no
C       need for the next commands
        IF(ISTOP.EQ.1) GOTO 25
```

```
C       update A-hrs
        II=JSUB(JTOP,JLAST(J),JDIF)
        CALL JJCVT(JDIF)
        CALL CVTTIM(JDIF,IHR,IMIN,ISCND,ITICK)
        HR=FLOAT(IHR)+FLOAT(IMIN)/60.+FLOAT(ISCND)/3600.+FLOAT(ITICK)/
        1216000.
        READCH=(RLOGG(A(19,J),A(20,J),8,0)-OFF(J))*VAL(3,J)
        Q(A(3,J)+1,J)=Q(A(3,J)+1,J)+HR*ABS(READCH)
25      ISTOP=0
C       convert V-hrs to A-hrs, and store them in BUF area
        BUF(125+A(3,J),J)=Q(A(3,J)+1,J)*A(10,J)/5.
        WRITE(6,*) 'CELL ',J,' , A-HRS PASSED : ',BUF(125+A(3,J),J)
        IF(A(3,J).EQ.0) WRITE(6,*) 'ON DISCHARGE, CYCLE',ICYCLE(J)
        IF(A(3,J).EQ.1) WRITE(6,*) 'ON CHARGE, CYCLE',ICYCLE(J)
        REWIND 6
C       if no cells in series, jump
        IF(B(A(3,J)+3,J).NE.3) GOTO 27
C       if companion cell still in active mode, force it to go in open
C       circuit
        IF(A(2,A(11,J)).EQ.1) II=JMOV(JTOP,JT(3,A(11,J)))
C       initialize A-hrs for the next half-cycle
27      Q(2-A(3,J),J)=0.
C       if single half-cycle mode, return
        IF(B(3+A(3,J),J).EQ.2.OR.B(3+A(3,J),J).EQ.5) RETURN
C       if cell was on charge and is put out of operation, apply trick,
C       so that, when it re-starts (on charge), the right number of
C       A-hrs to pass are defined
        IF(A(3,J).EQ.1.AND.A(1,J).EQ.0) Q(1,J)=QO(2,J)/VAL(4,J)
        RETURN
        END
```

```
      SUBROUTINE CVLC(J,ISEC,ISTOP,BUF)
C     THIS SUBROUTINE ORGANIZES CONSTANT CURRENT - LIMITED VOLTAGE CONTROL
      COMMON/PASS/A,B,JT,VAL,QO,Q
      COMMON/CURTIM/JLAST,JREM
      COMMON/CURCUR/CUR,COMP
      COMMON/CYCLCT/ICYCLE(16),MCYCLE(16)
      COMMON/TIMTIM/JCHECK,JSET,JCVLC
      COMMON/RES/JTOP,JMAX
      COMMON/OFFSET/OFF(16)
      INTEGER A(22,16),B(14,16)
      INTEGER*4 JT(10,16),JLAST(16),JREM(16),JDIF,JTOP,JCVLC,
     1JCHECK,JSET
      DIMENSION VAL(14,16),QO(2,16),Q(2,16),CUR(16),COMP(16)
      DIMENSION BUF(128,16),STTMP(16),X(16)
      LOGICAL*1 ANS
```

```
C               WHEN RUN IN FB, A COMMON FOR STTMP() NEEDS TO BE ADDED !!!
C               WHEN RUN IN FB, A COMMON FOR X() NEEDS TO BE ADDED !!!
C       get time of day
        CALL GTIM(JTOP)
        CALL JJCVT(JTOP)
        IFLAG=0
        GOTO(5,10,65) ISEC
C
C               initialization section
C
C       if not single mode, update A-hrs to pass on charge
5       IF(B(4,J).NE.5) QO(2,J)=VAL(4,J)*Q(1,J)
C       initialize slope for voltage regulation
        X(J)=1.
D       TYPE 900
900     FORMAT(1X,'CVLC INIT')
C       if no cells in series, return
        IF(B(4,J).NE.6) RETURN
C       if companion cell still in open circuit, force it to start a
C       charge
        IF(A(2,A(11,J)).EQ.0) II=JMOV(JTOP,JT(3,A(11,J)))
        RETURN
C
C               control section
C
10      CONTINUE
D       TYPE 1000
1000    FORMAT(1X,'CVLC  CONTROL')
C       if not new mode, jump
        IF(A(8,J).NE.0) GOTO 12
```

```
C
C       subsection : 1st pass
C
C       define current to be set
        CUR(J)=VAL(12,J)*VAL(3,J)*A(10,J)/5.
C       update action timer and JLAST timer
        II=JADD(JTOP,JCVLC,JT(2,J))
        II=JMOV(JTOP,JLAST(J))
C       if cell not controlling, jump
        IF(J.LT.A(11,J)) RETURN
C       define setpoint, call CALIB
        SETPNT=-VAL(12,J)/VAL(3,J)
        CALL CALIB(SETPNT,CUR(J),A(9,J),A(10,J),A(19,J),A(20,J),
        1OFF(J),.001,.005,A(8,J),ANS)
C       if ANS = F, set termination flag for this cell (and eventually
C       for its companion)
        IF(ANS.EQ.'F') A(1,J)=2
        IF(ANS.EQ.'F'.AND.A(11,J).NE.0) A(1,A(11,J))=2
C       if there is a companion cell, define its CUR
        IF(B(3,J).EQ.6) CUR(A(11,J))=CUR(J)
        RETURN
```

```
C       if this is the second pass through the control section, execute
C       the next commands; otherwise jump
12      IF(A(8,J).NE.1) GOTO 20
C
C       subsection: 2nd pass
C
C       start by saying that this cell is controlling
        A(8,J)=2
C       if there is no companion cell, no need to worry more about which
C       is controlling
        IF(B(3,J).NE.6) GOTO 20
C       if companion cell is controlling, jump
        IF(J.LT.A(11,J)) GOTO 15
C       companion cell is not controlling
        A(8,A(11,J))=4
        GOTO 20
C       companion cell is controlling, inverse flags
15      A(8,J)=4
        A(8,A(11,J))=2
C
C       subsection: A-hr update
C
C       compute A-hrs for this cell and (eventually) for its companion.
C       If the cell is not controlling, just get the time and read the
C       current : this will be useful if the cells switch in their
C       control
20      II=JSUB(JTOP,JLAST(J),JDIF)
        CALL JJCVT(JDIF)
        CALL CVTTIM(JDIF,IHR,IMIN,ISCND,ITICK)
        HR=FLOAT(IHR)+FLOAT(IMIN)/60.+FLOAT(ISCND)/3600.+FLOAT(ITICK)/
```

```
      1216000.
      READC=(RLOGG(A(19,J),A(20,J),8,0)-OFF(J))*VAL(3,J)
      IF(A(8,J).EQ.4) GOTO 25
      Q(2,J)=Q(2,J)+HR*ABS(READC)
      TYPE *,'CELL ',J,', A-HRS PASSED  ',Q(2,J)*A(10,J)/5.
      IF(B(3,J).NE.6) GOTO 25
      Q(2,A(11,J))=Q(2,A(11,J))+HR*ABS(READC)
      TYPE *,'CELL ',A(11,J),', A-HRS PASSED  ',Q(2,A(11,J))*
     1A(10,A(11,J))/5.
C     if A-hrs exceeded, set ISTOP flag, otherwise jump
25    IF(Q(2,J).LT.Q0(2,J)) GOTO 30
      ISTOP=1
      RETURN
C     if cell is already controlled on the voltage, jump
30    IF(A(8,J).EQ.3) GOTO 55
C
C     subsection: current control
C
C     if cell is not controlling, jump
      IF(A(8,J).EQ.4) GOTO 35
```

```
C       define setpoint and call CALIB
        SETPNT=-VAL(12,J)/VAL(3,J)
        CALL CALIB(SETPNT,CUR(J),A(9,J),A(10,J),A(19,J),A(20,J),
        1OFF(J),.001,.005,A(8,J),ANS)
C       if ANS = F, set termination flag for this cell (and eventually
C       for its companion)
        IF(ANS.EQ.'F') A(1,J)=2
        IF(ANS.EQ.'F'.AND.A(11,J).NE.0) A(1,A(11,J))=2
C       read the voltage on the desired channel for voltage regulation
35      READV=RLOGG(A(11+2*B(14,J),J),A(12+2*B(14,J),J),8,0)
C       if voltage close to be exceeded, jump to the section where the
C       cell is defined as regulated on the voltage
        IF(READV.GE.(VAL(14,J)*(1-.002))) GOTO 40
C       store time interval for current control in JDIF
        II=JMOV(JT(8,J),JDIF)
C       regulation on the current terminated, jump
        GOTO 60
C
C       subsection: change from constant current to constant voltage mode
C
C       if this cell was not controlling on current, set IFLAG
40      IF(A(8,J).EQ.4) IFLAG=1
C       cell is now controlling on voltage
        A(8,J)=3
C       if the current channel was watched for data taking rate
C       adjustment, jump
        IF(B(2,J).EQ.4) GOTO 45
C       otherwise, save the old specs for rate adjsutment, and now,
C       watch the current channel, with a step of 20 mA
        B(12,J)=B(2,J)
```

```
      B(2,J)=4
      STTMP(J)=VAL(2,J)
      VAL(2,J)=.020*5./A(10,J)
C     if no companion cell, jump
45    IF(B(4,J).NE.6) GOTO 55
C     companion cell is not controlling anymore
      A(8,A(11,J))=4
C     if for the companion cell, the current channel was watched for
C     adjustment of the data-taking rate, reset the old specs
      IF(B(2,A(11,J)).EQ.4) GOTO 50
      B(12,A(11,J))=B(2,A(11,J))
      B(2,A(11,J))=4
      STTMP(A(11,J))=VAL(2,A(11,J))
      VAL(2,A(11,J))=VAL(2,J)
C     if IFLAG was set, then the cell just became controlling : before,
C     its comapnion was controlling; update its A-hrs; otherwise, jump
```

```
50      IF(IFLAG.NE.1) GOTO 55
        Q(2,J)=Q(2,J)+HR*ABS(READC)
        TYPE *,'CELL ',J,', A-HRS PASSED  ',Q(2,J)*A(10,J)/5.
        Q(2,A(11,J))=Q(2,A(11,J))+HR*ABS(READC)
        TYPE *,'CELL ',A(11,J),', A-HRS PASSED  ',Q(2,A(11,J))*
       1A(10,A(11,J))/5.
        IF(Q(2,J).LT.Q0(2,J)) GOTO 55
        ISTOP=1
        RETURN
C
C       subsection: voltage control
C
C       define setpoint and call CALIBV
55      SETPNT=VAL(14,J)
        CALL CALIBV(SETPNT,CUR(J),A(9,J),A(10,J),A(19,J),A(20,J),
       1A(11+2*B(14,J),J),A(12+2*B(14,J),J),OFF(J),X(J),.001,.005,ANS)
C       if ANS = F, set termination flag for this cell (and eventually
C       for its companion)
        IF(ANS.EQ.'F') A(1,J)=2
        IF(ANS.EQ.'F'.AND.A(11,J).NE.0) A(1,A(11,J))=2
C       put 15 seconds in JDIF
C
C       subsection: timer update
C
        II=JMOV(JCVLC,JDIF)
C       update action timer with JDIF
60      II=JADD(JTOP,JDIF,JT(2,J))
C       if cell not controlling, return
        IF(A(8,J).EQ.4) RETURN
C       update JLAST for this cell and eventually for its companion
```

```
        II=JMOV(JTOP,JLAST(J))
        IF(B(3,J).NE.6) RETURN
        II=JMOV(JTOP,JLAST(A(11,J)))
C       update CUR for the companion
        CUR(A(11,J))=CUR(J)
        RETURN
C
C               termination section
C
65      CONTINUE
D       TYPE 1100
1100    FORMAT(1X,'CVLC TERMIN')
C       if necessary, re-store the old specs for data-taking rate adjustment
        IF(B(2,J).NE.4) GOTO 70
        B(2,J)=B(12,J)
        VAL(2,J)=STTMP(J)
C       if ISTOP was set, no need for next commands
```

```
70      IF(ISTOP.EQ.1) GOTO 75
        II=JSUB(JTOP,JLAST(J),JDIF)
        CALL JJCVT(JDIF)
        CALL CVTTIM(JDIF,IHR,IMIN,ISCND,ITICK)
        HR=FLOAT(IHR)+FLOAT(IMIN)/60.+FLOAT(ISCND)/3600.+FLOAT(ITICK)/
        1216000.
        READC=(RLOGG(A(19,J),A(20,J),8,0)-OFF(J))*VAL(3,J)
        Q(2,J)=Q(2,J)+HR*ABS(READC)
75      ISTOP=0
C       convert V-hrs to A-hrs, and store them in the BUF area
        BUF(125+A(3,J),J)=Q(2,J)*A(10,J)/5.
        WRITE(6,*) 'CELL ',J,' , A-HRS PASSED :',BUF(125+A(3,J),J)
        WRITE(6,*) 'ON CHARGE, CYCLE',ICYCLE(J)
        REWIND 6
C       if no companion cell, jump
        IF(B(4,J).NE.6) GOTO 80
C       if companion cell still in active mode, force it to go in
C       open circuit
        IF(A(2,A(11,J)).EQ.1) II=JMOV(JTOP,JT(3,A(11,J)))
C       initialize A-hrs for next half-cycle
80      Q(2-A(3,J),J)=0.
C       if single half-cycle charge, return
        IF(B(4,J).EQ.5) RETURN
C       trick, see CC subroutine
        IF(A(1,J).EQ.0) Q(1,J)=QO(2,J)/VAL(4,J)
        RETURN
        END
```

```
      SUBROUTINE CP(J,ISEC,ISTOP,BUF)
C     THIS SUBROUTINE ORGANIZES CONSTANT POWER CONTROL
      COMMON/PASS/A,B,JT,VAL,QO,Q
      COMMON/CURTIM/JLAST,JREM
      COMMON/CURCUR/CUR,COMP
      COMMON/CYCLCT/ICYCLE(16),MCYCLE(16)
      COMMON/TIMTIM/JCHECK,JSET,JCVLC
      COMMON/RES/JTOP,JMAX
      COMMON/OFFSET/OFF(16)
      INTEGER A(22,16),B(14,16)
      INTEGER*4 JT(10,16),JLAST(16),JREM(16),JDIF,JTOP,JCHECK,JSET,
     1JCVLC
      DIMENSION VAL(14,16),QO(2,16),Q(2,16),CUR(16),COMP(16)
      DIMENSION BUF(128,16),X(16)
      LOGICAL*1 ANS
C           WHEN RUN IN FB, A COMMON FOR X( ) NEEDS TO BE ADDED !!!
```

```
C       get time of day
        CALL GTIM(JTOP)
        CALL JJCVT(JTOP)
        GOTO(5,10,40) ISEC
C
C               initialization section
C
C       initialize slope for power regulation
5       X(J)=-.1
D       TYPE 900
900     FORMAT(1X,'CP INIT')
C       if no cells in series, jump
        IF(B(3,J).NE.6) RETURN
C       if companion cell still in open circuit, force it to start a
C       discharge
        IF(A(2,A(11,J)).EQ.0) II=JMOV(JTOP,JT(3,A(11,J)))
        RETURN
C
C               control section
C
10      CONTINUE
D       TYPE 1000
1000    FORMAT(1X,'CP CONTROL')
C       update action timer (with 15 seconds)
        II=JADD(JTOP,JCVLC,JT(2,J))
c       if not new mode, jump
        IF(A(8,J).NE.0) GOTO 15
C
C       subsection: first pass
C
```

```
      CUR(J)=.2
c     update JLAST
      II=JMOV(JTOP,JLAST(J))
C     highest cell controls at 1st pass
      IF(J.LT.A(11,J)) RETURN
C     how about getting ourselves started on the regulation...
C     charge for 3 ticks, @ .2 A; then discharge the same amount
      CALL SET(1,1,A(9,J))
      CALL IMPOSE(A(9,J),.2,A(10,J),IDUM)
      CALL ISLEEP(0,0,0,3)
      CALL IMPOSE(A(9,J),0.,A(10,J),IDUM)
      CALL SET(0,0,A(9,J))
      CALL SET(1,0,A(9,J))
      CALL IMPOSE(A(9,J),.2,A(10,J),IDUM)
      CALL ISLEEP(0,0,0,3)
C     jump to power control subsection
      GOTO 35
C
C     subsection: 2nd and later passes
C
C     if no companion cell, no worry about which one is controlling
15    IF(B(3,J).NE.6) GOTO 25
```

```
C       measure the voltages on the desired channels, for the 2 cells
        ICHV1=A(11+2*B(13,J),J)
        IRV1=A(12+2*B(13,J),J)
        ICHV2=A(11+2*B(13,A(11,J)),A(11,J))
        IRV2=A(12+2*B(13,A(11,J)),A(11,J))
C       compute difference between this cell's and its companion's voltage
        DELTAV=RLOGG(ICHV1,IRV1,8,0)-RLOGG(ICHV2,IRV2,8,0)
C       if this cell's voltage is lower, it may not control
        IF(DELTAV.LE.(-.010)) GOTO 20
        IF(DELTAV.GE.(.010)) GOTO 25
C       if not well defined (+/- 10 mV), then highest cell controls
        IF(J.GT.A(11,J)) GOTO 25
C       if A-hrs exceeded, set ISTOP flag
20      IF(Q(1,J).GE.Q0(1,J)) ISTOP=1
        RETURN
C
C       subsection: A-hr update
C
C       update JLAST timer and A-hrs passed for this cell (and eventually
C       for its companion)
25      II=JSUB(JTOP,JLAST(J),JDIF)
        CALL JJCVT(JDIF)
        CALL CVTTIM(JDIF,IHR,IMIN,ISCND,ITICK)
        HR=FLOAT(IHR)+FLOAT(IMIN)/60.+FLOAT(ISCND)/3600.+FLOAT(ITICK)/
        1216000.
        READCH=(RLOGG(A(19,J),A(20,J),8,0)-OFF(J))*VAL(3,J)
        Q(1,J)=Q(1,J)+HR*ABS(READCH)
        TYPE *,'CELL ',J,', A-HRS PASSED  ',Q(1,J)*A(10,J)/5.
        IF(B(3,J).NE.6) GOTO 30
        Q(1,A(11,J))=Q(1,A(11,J))+HR*ABS(READCH)
```

```
      TYPE *,'CELL ',A(11,J),', A-HRS PASSED  ',Q(1,A(11,J))*
     1A(10,A(11,J))/5.
      II=JMOV(JTOP,JLAST(A(11,J)))
30    II=JMOV(JTOP,JLAST(J))
C     if A-hrs exceeded, set ISTOP flag
      IF(Q(1,J).LT.Q0(1,J)) GOTO 35
      ISTOP=1
      RETURN
C
C     subsection: power control
C
C     define setpoint, call CALIBP
35    SETPNT=VAL(13,J)/VAL(3,J)
      CALL CALIBP(SETPNT,CUR(J),A(9,J),A(10,J),A(19,J),A(20,J),
     1A(11+2*B(13,J),J),A(12+2*B(13,J),J),OFF(J),X(J),.010,ANS)
C     if ANS = F, set termination flag for this cell, and eventually
C     for its companion
      IF(ANS.EQ.'F') A(1,J)=2
      IF(ANS.EQ.'F'.AND.A(11,J).NE.0) A(1,A(11,J))=2
```

```
C       if there is a companion cell, update its CUR
        IF(B(3,J).EQ.6) CUR(A(11,J))=CUR(J)
        RETURN
C
C               termination section
C
40      CONTINUE
D       TYPE 1100
1100    FORMAT(1X,'CP TERMIN')
C       if ISTOP was set, no need for the next commands
        IF(ISTOP.EQ.1) GOTO 45
C       update A-hrs
        II=JSUB(JTOP,JLAST(J),JDIF)
        CALL JJCVT(JDIF)
        CALL CVTTIM(JDIF,IHR,IMIN,ISCND,ITICK)
        HR=FLOAT(IHR)+FLOAT(IMIN)/60.+FLOAT(ISCND)/3600.+FLOAT(ITICK)/
        1216000.
        READCH=(RLOGG(A(19,J),A(20,J),8,0)-OFF(J))*VAL(3,J)
        Q(1,J)=Q(1,J)+HR*ABS(READCH)
45      ISTOP=0
C       convert V-hrs to A-hrs and store it in BUF area
        BUF(125+A(3,J),J)=Q(1,J)*A(10,J)/5.
        WRITE(6,*) 'CELL ',J,' , A-HRS PASSED : ',BUF(125+A(3,J),J)
        IF(A(3,J).EQ.0) WRITE(6,*) 'ON DISCHARGE, CYCLE',ICYCLE(J)
        IF(A(3,J).EQ.1) WRITE(6,*) 'ON CHARGE, CYCLE',ICYCLE(J)
        REWIND 6
C       if no companion cell, jump
        IF(B(3,J).NE.6) GOTO 50
C       if companion cell still in active mode, force it to go in
C       open-circuit
```

```
      IF(A(2,A(11,J)).EQ.1) II=JMOV(JTOP,JT(3,A(11,J)))
C     initialize A-hrs for next half-cycle
50    Q(2-A(3,J),J)=0.
      RETURN
      END


      SUBROUTINE EXCON(J,ISEC,ISTOP,BUF)
C     THIS SUBROUTINE ORGANIZES THE PULSED CURRENT CONTROL
      COMMON/PASS/A,B,JT,VAL,QO,Q
      COMMON/CURTIM/JLAST,JREM
      COMMON/CYCLCT/ICYCLE(16),MCYCLE(16)
      COMMON/RES/JTOP,JMAX
      COMMON/OFFSET/OFF(16)
      INTEGER A(22,16),B(14,16)
      INTEGER*4 JT(10,16),JLAST(16),JREM(16),JDIF,JTOP,J230
      DIMENSION VAL(14,16),QO(2,16),Q(2,16),BUF(128,16)
```

```
C       get time of day
        CALL GTIM(JTOP)
        CALL JJCVT(JTOP)
        GOTO(5,10,20) ISEC
C
C               initialization section
C
C       if single half-cycle mode, jump
5       IF(B(3+A(3,J),J).EQ.8) GOTO 6
C       define A-hrs to pass on charge
        IF(A(3,J).EQ.1) QO(2,J)=VAL(4,J)*Q(1,J)
6       CONTINUE
D       TYPE 900
900     FORMAT(1X,'EXCON INIT')
C       if no cell in series, return
        IF(B(A(3,J)+3,J).NE.9) RETURN
C       if companion cell still in open-circuit, force it to start a
C       charge or discharge
        IF(A(2,A(11,J)).EQ.0) II=JMOV(JTOP,JT(3,A(11,J)))
        RETURN
C
C               control section
C
10      CONTINUE
D       TYPE 1000
1000    FORMAT(1X,'EXCON  CONTROL')
C       if not new mode, jump
        IF(A(8,J).NE.0) GOTO 11
C
C       subsection: 1st pass
```

```
C
C       define INT*4 timer = 2 minutes & 30 seconds
        CALL JTIME(0,2,30,0,J230)
        CALL JJCVT(J230)
C       update action timer
        II=JADD(JTOP,J230,JT(2,J))
C       switch from regular A/D current channel to A/D channel
C       connected to averager circuit (always 1 higher !)
        A(19,J)=A(19,J)+1
C       if controlling cell, set the controller in external mode
        IF(J.GT.A(11,J)) CALL EXT(1,A(9,J))
        GOTO 14
C
C       subsection: 2nd and later passes
C
C       update action timer
11      II=JADD(JTOP,JT(8,J),JT(2,J))
C       if cell not controlling, jump
        IF(J.LT.A(11,J)) GOTO 13
```

```
C       update A-hrs for this cell (and eventually for companion)
        II=JSUB(JTOP,JLAST(J),JDIF)
        CALL JJCVT(JDIF)
        CALL CVTTIM(JDIF,IHR,IMIN,ISCND,ITICK)
        HR=FLOAT(IHR)+FLOAT(IMIN)/60.+FLOAT(ISCND)/3600.+FLOAT(ITICK)/
        1216000.
        READCH=(RLOGG(A(19,J),A(20,J),8,0)-OFF(J))*VAL(3,J)
        Q(A(3,J)+1,J)=Q(A(3,J)+1,J)+HR*ABS(READCH)
        TYPE *,'CELL ',J,', A-HRS PASSED  ',Q(A(3,J)+1,J)*A(10,J)/5.
        IF(B(3+A(3,J),J).NE.9) GOTO 12
        Q(A(3,A(11,J))+1,A(11,J))=Q(A(3,A(11,J))+1,A(11,J))+HR*ABS(READCH)
        TYPE *,'CELL ',A(11,J),', A-HRS PASSED  ',Q(A(3,A(11,J))+1,
        1A(11,J))*A(10,A(11,J))/5.
C       update JLAST timers
        II=JMOV(JTOP,JLAST(A(11,J)))
12      II=JMOV(JTOP,JLAST(J))
C       if A-hrs not exceeded, return
13      IF(Q(A(3,J)+1,J).LT.Q0(A(3,J)+1,J)) RETURN
C       set ISTOP flag and return
        ISTOP=1
        RETURN
C       update JLAST timer
14      II=JMOV(JTOP,JLAST(J))
        RETURN
C
C               termination section
C
20      CONTINUE
D       TYPE 1100
1100    FORMAT(1X,'EXCON TERMIN')
```

```
C       if ISTOP flag set, no need for next commands
        IF(ISTOP.EQ.1) GOTO 25
C       update A-hrs
        II=JSUB(JTOP,JLAST(J),JDIF)
        CALL JJCVT(JDIF)
        CALL CVTTIM(JDIF,IHR,IMIN,ISCND,ITICK)
        HR=FLOAT(IHR)+FLOAT(IMIN)/60.+FLOAT(ISCND)/3600.+FLOAT(ITICK)/
        1216000.
        READCH=(RLOGG(A(19,J),A(20,J),8,0)-OFF(J))*VAL(3,J)
        Q(A(3,J)+1,J)=Q(A(3,J)+1,J)+HR*ABS(READCH)
25      ISTOP=0
C       convert V-hrs to A-hrs and store in BUF area
        BUF(125+A(3,J),J)=Q(A(3,J)+1,J)*A(10,J)/5.
        WRITE(6,*) 'CELL ',J,' , A-HRS PASSED : ',BUF(125+A(3,J),J)
        IF(A(3,J).EQ.0) WRITE(6,*) 'ON DISCHARGE, CYCLE',ICYCLE(J)
        IF(A(3,J).EQ.1) WRITE(6,*) 'ON CHARGE, CYCLE',ICYCLE(J)
        REWIND 6
C       re-store regular A/D channel for current
        A(19,J)=A(19,J)-1
```

```
C       if no companion cell, jump
        IF(B(A(3,J)+3,J).NE.9) GOTO 27
C       if companion cell is still in active mode, force it to go in
C       open circuit
        IF(A(2,A(11,J)).EQ.1) II=JMOV(JTOP,JT(3,A(11,J)))
C       initialize A-hrs of next half-cycle
27      Q(2-A(3,J),J)=0.
C       if single half-cycle mode, return
        IF(B(3+A(3,J),J).EQ.8) RETURN
C       trick, see CC subroutine
        IF(A(3,J).EQ.1.AND.A(1,J).EQ.0) Q(1,J)=Q0(2,J)/VAL(4,J)
        RETURN
        END



        SUBROUTINE CALIB(SETPNT,CON,ICC,IFULL,ICHAN,IRAN,OFF,
        1TOLMIN,TOLMAX,IBEG,ANS)
C       THIS SUBROUTINE REGULATES THE CURRENT THROUGH A CELL
        LOGICAL*1 ANS
C       define optimal values
        CONOP=CON
        READOP=0.
        ANS='Y'
        ICNT=-1
        DO 5 K=1,11
        ICNT=ICNT+1
C       if first pass in DO loop, jump
```

```
      IF(K.EQ.1) GOTO 3
C     if current to be set is out of bounds, error
      IF(CON.GT.10..OR.CON.LT.0.) GOTO 15
C     set current
      CALL IMPOSE(ICC,CON,IFULL,IDUM)
C     sleep for 3 tickS
      CALL ISLEEP(0,0,0,3)
C     read current
3     READCH=RLOGG(ICHAN,IRAN,8,0)-OFF
C     if first pass through the subroutine, since the start of the
C     half-cycle, jump
      IF(IBEG.NE.0) GOTO 4
C     if current to be set is out of bounds, error
      IF(CON.GT.10..OR.CON.LT.0.) GOTO 15
C     set current
      CALL IMPOSE(ICC,CON,IFULL,IDUM)
C     sleep for 3 ticks
      CALL ISLEEP(0,0,0,3)
```

```
C       read current
        READCH=RLOGG(ICHAN,IRAN,8,0)-OFF
        GOTO 10
C       if current within the minimum tolerance of SETPNT, done
4       IF(ABS(SETPNT-READCH).LE.TOLMIN) GOTO 10
C       if current is not best one so far, jump
        IF(ABS(SETPNT-READCH).GE.ABS(SETPNT-READOP)) GOTO 45
C       define optimal current setpoint and current
        CONOP=CON
        READOP=READCH
C       if no current through current controller, error
45      IF(ABS(READCH).LT.1.E-04) GOTO 15
C       iterate
        CON=CON*SETPNT/READCH
C       if current to be set is out of bounds, error
        IF(CON.GT.10..OR.CON.LT.0.) GOTO 15
5       CONTINUE
C       if current is not within the maximum tolerance of SETPNT, ANS = N
        IF(ABS(SETPNT-READOP).GT.TOLMAX) ANS='N'
C       remember the best current reading
        READCH=READOP
C       remember the best current setpoint
        CON=CONOP
C       if current to be set is out of bounds, error
        IF(CON.GT.10..OR.CON.LT.0.) GOTO 15
C       set best current
        CALL PLACE(ICC,CON,IFULL,IDUM)
10      CONTINUE
D       TYPE 1000,ANS,ICNT,(READCH*IFULL/5.)
1000    FORMAT(1X,A1,3X,I2,3X,F10.5)
```

```
      RETURN
C     error
15    ANS='F'
      WRITE(6,*) 'CONTROLLER ',ICC,': CANNOT REGULATE ON THE CURRENT'
      REWIND 6
C     set current to 0
      CALL IMPOSE(ICC,0.,IFULL,IDUM)
      RETURN
      END


      SUBROUTINE CALIBV(SETPNT,CUR,ICC,IFULL,ICHAMP,IRAMP,ICHV,IRV,
     1OFF,X,TOLMIN,TOLMAX,ANS)
C     THIS SUBROUTINE REGULATES THE VOLTAGE OF A CELL
      LOGICAL*1 ANS
```

```
C       read current (and convert it) and voltage
        AMPNEW=(RLOGG(ICHAMP,IRAMP,8,0)-OFF)*IFULL/5.
        VNEW=RLOGG(ICHV,IRV,8,0)
        ICNT=-1
C       define best current setpoint and best voltage
        VOPT=0.
        CUROPT=CUR
        ANS='Y'
        DO 20 K=1,11
        ICNT=ICNT+1
C       if first DO loop pass, jump
        IF(K.EQ.1) GOTO 10
C       iterate
        CUR=CUR+(SETPNT-VOLD)/X
C       if current setpoint is out of bounds, error
        IF(CUR.GT.10..OR.CUR.LT.0.) GOTO 30
C       set current
        CALL IMPOSE(ICC,CUR,IFULL,IDUM)
C       sleep for 3 ticks
        CALL ISLEEP(0,0,0,3)
C       read current, convert
        AMPNEW=(RLOGG(ICHAMP,IRAMP,8,0)-OFF)*IFULL/5.
C       read voltage
        VNEW=RLOGG(ICHV,IRV,8,0)
C       if new and old current almost the same, skip next commands
        IF(ABS(AMPNEW-AMPOLD).LE.1.E-02) GOTO 10
C       update slope for regulation
        X=(VNEW-VOLD)/(AMPNEW-AMPOLD)
C       if bad, set back to 1
        IF(X.LE.0) X=1.
```

```
C       if voltage within the minimum tolerance of SETPNT, done
10      IF(ABS(SETPNT-VNEW).LE.TOLMIN) GOTO 25
C       define old values
        AMPOLD=AMPNEW
        VOLD=VNEW
C       if voltage not best so far, jump
        IF(ABS(SETPNT-VNEW).GE.ABS(SETPNT-VOPT)) GOTO 20
C       define optimal values
        VOPT=VNEW
        CUROPT=CUR
20      CONTINUE
C       remember best voltage
        VNEW=VOPT
C       remember best current setpoint
        CUR=CUROPT
C       if current to be set is out of bounds, error
        IF(CUR.GT.10..OR.CUR.LT.0.) GOTO 30
C       set current
        CALL IMPOSE(ICC,CUR,IFULL,IDUM)
C       if voltage not within the maximum tolerance of SETPNT, ANS = N
        IF(ABS(SETPNT-VNEW).GT.TOLMAX) ANS='N'
25      CONTINUE
D       TYPE 1000,ANS,ICNT,VNEW,X
1000    FORMAT(1X,A1,3X,I2,3X,F10.5,3X,F10.5)
        RETURN
```

```
C       error
30      ANS='F'
        WRITE(6,*) 'CONTROLLER ',ICC,': CANNOT REGULATE ON THE VOLTAGE'
        REWIND 6
C       set current 0
        CALL IMPOSE(ICC,0.,IFULL,IDUM)
        RETURN
        END


        SUBROUTINE CALIBP(SETPNT,CUR,ICC,IFULL,ICHAMP,IRAMP,ICHV,IRV,
        1OFF,X,TOLMAX,ANS)
C       THIS SUBROUTINE REGULATES THE POWER THROUGH A CELL
        LOGICAL*1 ANS
C       read current and convert
        AMPNEW=(RLOGG(ICHAMP,IRAMP,8,0)-OFF)*IFULL/5.
C       read voltage
        VNEW=RLOGG(ICHV,IRV,8,0)
C       compute power
        PNEW=AMPNEW*VNEW
        ICNT=-1
C       define optimal values
        POPT=0.
        CUROPT=CUR
        ANS='Y'
        DO 20 K=1,11
        ICNT=ICNT+1
```

```
C       if 1st pass of DO loop, jump
        IF(K.EQ.1) GOTO 10
C       compute denominator for iteration expression
        DENOM=VOLD+X*AMPOLD
C       if very small, error
        IF(ABS(DENOM).LT.1.E-04) GOTO 30
C       iterate
        CUR=CUR+(SETPNT-POLD)/DENOM
C       if current setpoint is out of bounds, error
        IF(ABS(CUR).GT.2*ABS(FLOAT(IFULL))) GOTO 30
C       set current
        CALL IMPOSE(ICC,CUR,IFULL,IDUM)
C       sleep for 3 ticks
        CALL ISLEEP(0,0,0,3)
C       read current and convert, read voltage, compute power
        AMPNEW=(RLOGG(ICHAMP,IRAMP,8,0)-OFF)*IFULL/5.
        VNEW=RLOGG(ICHV,IRV,8,0)
        PNEW=AMPNEW*VNEW
C       if new and old currents almost the same, skip next commands
        IF(ABS(AMPNEW-AMPOLD).LE.1.E-02) GOTO 10
```

```
C       update slope for iteration
        X=(VNEW-VOLD)/(AMPNEW-AMPOLD)
C       if no good, set back to -.1
        IF(X.GT.0.) X=-.1
C       define precision on a power measurement (.5 mV on voltage,
C       1 mA on current)
10      ACC=.0005*AMPNEW+.001*VNEW
C       if power within precision range of SETPNT, done
        IF(ABS(SETPNT-PNEW).LE.ACC) GOTO 25
C       define old values
        POLD=PNEW
        AMPOLD=AMPNEW
        VOLD=VNEW
C       if not best power so far, jump
        IF(ABS(SETPNT-PNEW).GE.ABS(SETPNT-POPT)) GOTO 20
C       define optimal values
        POPT=PNEW
        CUROPT=CUR
20      CONTINUE
C       remember best power and best current setpoint
        PNEW=POPT
        CUR=CUROPT
C       if current setpoint is out of bounds, error
        IF(ABS(CUR).GT.2*ABS(FLOAT(IFULL))) GOTO 30
C       set current
        CALL IMPOSE(ICC,CUR,IFULL,IDUM)
C       if power is not within maximum tolerance of SETPNT,
C       ANS = N
        IF(ABS(SETPNT-PNEW).GT.TOLMAX) ANS='N'
C       reset X to -.1
```

```
        X=-.1
25      CONTINUE
D       TYPE 1000,ANS,ICNT,PNEW,X
1000    FORMAT(1X,A1,3X,I2,3X,F10.5,3X,F10.5)
        RETURN
C       error
30      ANS='F'
        WRITE(6,*) 'CONTROLLER ',ICC,': CANNOT REGULATE ON THE POWER'
        REWIND 6
C       set current to 0
        CALL IMPOSE(ICC,0.,IFULL,IDUM)
        RETURN
        END
```

```
      SUBROUTINE INPUT(J,IEX,IMODE,BUF)
C     THIS SUBROUTINE TAKES CARE OF DATA STORAGE AND LIMIT CHECKING
C     IMODE=0          NORMAL MODE
C     IMODE=1          WRITE BUF TO FILE, NO DATA, NO LIMITS
C     IMODE=2          DATA, NO LIMITS
C     IMODE=3          DATE & TIME, NO DATA, NO LIMITS
      COMMON/PASS/A,B,JT,VAL,QO,Q
      COMMON/CHANNELS/ICHAN(16)
      COMMON/RES/JTOP,JMAX
      COMMON/CYCLCT/ICYCLE(16),MCYCLE(16)
      COMMON/OFFSET/OFF(16)
      LOGICAL*1 ANS
      INTEGER A(22,16),B(14,16),ICOUNT(4)
      INTEGER*4 JT(10,16),JTOP,BB
      DIMENSION VAL(14,16),QO(2,16),Q(2,16)
      DIMENSION BUF(128,16)
      EQUIVALENCE(REAL1,ICOUNT(1))
      EQUIVALENCE(REAL2,ICOUNT(3))
D     TYPE *,'INPUT : MODE ',IMODE
      IEX=0
      ITAKE=1
      IF(IMODE.EQ.1) GOTO 27
      ANS='Y'
C     wait for I/O on the cell's channel to be completed
      II=IWAIT(ICHAN(J))
      ITAKE=0
C     update data storage counter
```

```
      A(6,J)=A(6,J)+1
C     update column number in BUF where storage of next data record
C     will start
      INDEX=(A(6,J)-1)*(A(12,J)+1)+1
C     get time
      CALL GTIM(JTOP)
      CALL JJCVT(JTOP)
      GOTO(1,1,102,101) IMODE+1
C     check if you must store data
1     CALL TIMEAS(ANS,J,BUF)
C     if you mustn't, you shan't
      IF(ANS.NE.'Y') ANS='N'
      GOTO 102
C     get date
101   CALL IDATE(ICOUNT(1),ICOUNT(2),ICOUNT(3))
C     if time to increment cycle #, please do it
      IF(MCYCLE(J).EQ.A(3,J)) ICYCLE(J)=ICYCLE(J)+1
      ICOUNT(4)=ICYCLE(J)
```

```
C       store date and cycle #
        BUF(INDEX,J)=REAL1
        INDEX=INDEX+1
        BUF(INDEX,J)=REAL2
        GOTO 25
C       if no data storage, jump
102     IF(ANS.EQ.'N') GOTO 2
C       convert JTOP to INT*4
        II=IAJFLT(JTOP,BUF(INDEX,J))
        TYPE *,'INPUT'
2       TYPE 1550,J
1550    FORMAT(1X,'CELL ',I2)
        DO 15 K=1,5
        L=K-1
C       if channel is programmed 0 0, skip it
        IF(A(14+2*L,J).EQ.0) GOTO 15
C       see previous INDEX update
        INDEX=INDEX+1
C       read an A/D channel
        BUF(INDEX,J)=RDLOGG(A(13+2*L,J),A(14+2*L,J),8,0,J)
C       if not current measurement, jump
        IF(K.NE.4) GOTO 3
C       go from V to A (50 mV shunt) and use calibration factor
        BUF(INDEX,J)=((BUF(INDEX,J)-OFF(J))/5.)*A(10,J)*VAL(3,J)
C       if in open-circuit, or if no-limit mode, or if data storage
C       but start of mode, jump
3       IF(A(2,J).EQ.0.OR.IMODE.EQ.2.OR.(ANS.EQ.'Y'.AND.A(8,J).EQ.0))
        1GOTO 15
        TYPE *,BUF(INDEX,J)
        DO 4 KK=1,3
```

```
C       if present channel number (1 to 5) has a limit assigned to
C       it, jmp to 5
        IF(K.EQ.IABS(B(4+KK+3*A(3,J),J))) GOTO 5
4       CONTINUE
        GOTO 15
C       MM is 5, 6 or 7 for ds., 8, 9 or 10 for ch.
5       MM=4+KK+3*A(3,J)
C       if upper limit, jump
        IF(B(MM,J).GT.0) GOTO 10
C       if lower limit exceeded, set IEX
        IF(BUF(INDEX,J).LT.VAL(MM,J)) IEX=1
        GOTO 15
C       if upper limit exceeded, set IEX
10      IF(BUF(INDEX,J).GT.VAL(MM,J)) IEX=1
15      CONTINUE
        TYPE 1900,A(12,J),J
1900    FORMAT(1X,'THERE ARE ',I2,' CHANNELS ACTIVE FOR CELL ',I2)
C       skip next commands if no limits exceeded
        IF(IEX.NE.1) GOTO 25
        TYPE 2000
```

```
2000    FORMAT(1X,'LIMIT EXCEEDED')
        DO 20 I=1,5
C       ring bell 5 * to announce that something is changing
17      IF(ITTOUR("007).NE.0) GOTO 17
20      CONTINUE
C       if data must be stored, jump
25      IF(ANS.NE.'N') GOTO 26
C       no data stored, reset counter
        A(6,J)=A(6,J)-1
        RETURN
C       decrement flag for data storage
26      A(4,J)=A(4,J)-1
C       M is 1 for ch, 2 for o-c, 3 for ds
        M=1+(1-A(2,J))+A(2,J)*(2-2*A(3,J))
C       if 1st data record (new block), set pointer in BUF pointer area
        IF(A(6,J).EQ.1) BUF(120+M,J)=1
C       if block (BUF) is full, have to write it to the file
        IF(A(6,J).GE.(240/(2*(A(12,J)+1)))) ITAKE=1
C       if flag for data storage > 1 and block not full, return
        IF(A(4,J).GT.1.AND.ITAKE.EQ.0) RETURN
C       if needed, open a data file for cell J
27      IF(A(5,J).EQ.0) CALL FILOPC(J,0)
C       reset data storage flag
        A(4,J)=3
C       write data
        II=IWRITE(256,BUF(1,J),A(7,J),ICHAN(J))
D       TYPE 9250,J
9250    FORMAT(1X,'I PUT DATA OF CELL`',I2,' ON THE APPROPRIATE FILE')
C       if block is not full, return
        IF(ITAKE.EQ.0) RETURN
```

```
C       update block number for next transfer
        A(7,J)=A(7,J)+1
C       re-initialize data-storage counter
        A(6,J)=0
C       initialize pointer and A-hr storage area (in BUF)
        DO 30 K=1,8
30      BUF(120+K,J)=0
C       get time
        CALL GTIM(JTOP)
        CALL JJCVT(JTOP)
        RETURN
        END
```

```
      SUBROUTINE TIMEAS(ANS,J,BUF)
C     THIS SUBROUTINE DECIDES ON WHETHER A DATA RECORD MUST BE STORED
C     OR NOT. IT ALSO REGULATES THE INTERVAL FOR DATA STORAGE AND
C     LIMIT CHECKING
      COMMON/PASS/A,B,JT,VAL,QO,Q
      COMMON/INT4/JTWO,JFOUR,JFAC
      COMMON/TIMTIM/JCHECK,JSET,JCVLC
      COMMON/CURTIM/JLAST,JREM
      COMMON/CURCUR/CUR,COMP
      COMMON/RES/JTOP,JMAX
      LOGICAL*1 ANS
      INTEGER*4 JT(10,16),JREM(16),JSET,JTOP,JBUF,JDIF,JNEXT,
     1JTWO,JFOUR,JCHECK,BB,JCVLC,JLAST(16),JFAC(16)
      INTEGER A(22,16),B(14,16)
      DIMENSION VAL(14,16),COMP(16),QO(2,16),Q(2,16),CUR(16)
      DIMENSION BUF(128,16)
D     TYPE 97
97    FORMAT(1X,'TIMEAS')
C     get time
      CALL GTIM(JTOP)
      CALL JJCVT(JTOP)
C     if not 1st pass through the routine since the beginning of this
C     half-cycle, jump
      IF(A(8,J).NE.0) GOTO 5
      A(8,J)=1
      ANS='Y'
C     get A/D channel and range for channel which is watched for
C     data storage rate adjustment
```

```
      ICH=A(13,J)*(1-A(2,J))+A(2,J)*A(13+2*(B(1+A(3,J),J)-1),J)
      IR=A(14,J)*(1-A(2,J))+A(2,J)*A(14+2*(B(1+A(3,J),J)-1),J)
C     read that channel
      TEST=RDLOGG(ICH,IR,8,0,J)
      IFAC=0
C     if o-c, jump
      IF(A(2,J).EQ.0) GOTO 25
C     divide DTinp by 2; if > or = 15 seconds, jump
1     II=JDIV(JT(10,J),JTWO,JT(10,J))
      IF(JCMP(JT(10,J),JCVLC).GE.0) GOTO 2
C     DTinp = 15 seconds
      II=JMOV(JCVLC,JT(10,J))
      GOTO 25
C     divide DTstore by JFAC
2     II=JDIV(JT(4,J),JFAC(J),JT(4,J))
      IFAC=IFAC+1
C     execute the previous commands a second time
      IF(IFAC.EQ.1) GOTO 1
      GOTO 25
```

```
C       find column in BUF where last data record stored ends
5       INDEX=(A(6,J)-1)*(A(12,J)+1)
C       if result is 0, then it was in previous block; it is still
C       stored in BUF, terminating at column 120
        IF(INDEX.EQ.0) INDEX=120
C       convert the time of that last data record to INT*4
        II=JAFIX(BUF((INDEX-A(12,J)),J),JBUF)
        II=JSUB(JTOP,JBUF,JDIF)
C       if the last stored time and the present time differ by less
C       than JREM, jump
        IF(JCMP(JDIF,JREM(J)).LT.0) GOTO 10
C       data must be stored
        ANS='Y'
        GOTO 15
C       data must not be stored
10      ANS='N'
C       get A/D channel and range for channel which is watched for
C       data storage rate adjustment
15      ICH=A(13,J)*(1-A(2,J))+A(2,J)*A(13+2*(B(1+A(3,J),J)-1),J)
        IR=A(14,J)*(1-A(2,J))+A(2,J)*A(14+2*(B(1+A(3,J),J)-1),J)
C       read that channel
        TEST=RDLOGG(ICH,IR,8,0,J)
C       tolerance is 30 mV at o-c or what user entered for ch or ds
        TOL=.030*(1-A(2,J))+A(2,J)*VAL(1+A(3,J),J)
C       if new (TEST) value within tolerance of old (COMP) value, jump
        IF(ABS(TEST-COMP(J)).LT.TOL) GOTO 20
C       if difference is more than twice the tolerance, data must be
C       stored
        IF(ABS(TEST-COMP(J)).GT.(2*TOL)) ANS='Y'
C       divide DTinp by 2; if > or = 15 seconds, jump
```

```
      II=JDIV(JT(10,J),JTWO,JT(10,J))
      IF(JCMP(JT(10,J),JCVLC).GE.0) GOTO 17
C     DTinp = 15 seconds
      II=JMOV(JCVLC,JT(10,J))
      GOTO 25
C     divide DTstore by JFAC
17    II=JDIV(JT(4,J),JFAC(J),JT(4,J))
C     put DTstore in JREM
      II=JMOV(JT(4,J),JREM(J))
      GOTO 25
C     if DTinp = 1 min., jump
20    IF(JCMP(JT(10,J),JSET).EQ.0) GOTO 25
C     multiply Dtinp by 2; if < or = 1 min., jump
      II=JMUL(JT(10,J),JTWO,JT(10,J))
      IF(JCMP(JT(10,J),JSET).LE.0) GOTO 21
C     DTinp = 1 min.
      II=JMOV(JSET,JT(10,J))
      GOTO 25
```

```
C       multiply DTstore by JFAC
21      II=JMUL(JT(4,J),JFAC(J),JT(4,J))
C       update action timer for subroutine INPUT
25      II=JADD(JTOP,JT(10,J),JT(1,J))
C       keep last value of TEST
        COMP(J)=TEST
D       TYPE 1000,ANS
1000    FORMAT(1X,A1)
C       if no storage, jump
        IF(ANS.EQ.'N') GOTO 30
C       put DTstore in JREM
        II=JMOV(JT(4,J),JREM(J))
        RETURN
C       add JREM to time of last data record stored
30      II=JADD(JBUF,JREM(J),JNEXT)
C       if that time < action timer for INPUT, update the latter
        IF(JCMP(JNEXT,JT(1,J)).LT.0) II=JMOV(JNEXT,JT(1,J))
        RETURN
        END




        SUBROUTINE FILOPC(IOP,ICLO)
C       THIS SUBROUTINE OPENS OR CLOSES A SCRATCH-DATA-FILE FOR CELL J
        COMMON/CHANNELS/ICHAN(16)
        COMMON/PASS/A,B,JT,VAL,QO,Q
        INTEGER A(22,16),B(14,16),DBLK(4)
        INTEGER*4 JT(10,16)
```

```
      DIMENSION VAL(14,16),QO(2,16),Q(2,16)
      LOGICAL*1 FNAME(3),GNAME(3)
      DATA DBLK/3RVOL,3RDAT,3ROO ,3RUNF/
C     which cell needs a file to be opened or closed ?
      J=IOP+ICLO
C     if need to close, jump
      IF(ICLO.NE.0) GOTO 10
      TYPE 99
99    FORMAT(1X,'FILOPC - OPEN')
C     encode the name of the file for cell J
      GNAME(1)=IOP/10
      GNAME(2)=MOD(IOP,10)
      GNAME(3)='0'
      ENCODE(3,100,FNAME)GNAME
100   FORMAT(2I1,A1)
      N=IRAD50(3,FNAME,DBLK(3))
      TYPE 900,ICHAN(J),(FNAME(KK),KK=1,3)
900   FORMAT(1X,'CHANNEL ',I2,3X'NAME ',3A1)
C     open a file that already exists
      TYPE 910
```

```
910     FORMAT(1X,'LOOKUP')
        II=LOOKUP(ICHAN(J),DBLK)
        IF(II.LT.0.AND.II.NE.-2) TYPE 350,II
        IF(II.GE.0) GOTO 5
C       if II is -2, then file does not exist: create a new one !
        TYPE 920
920     FORMAT(1X,'IENTER')
        II=IENTER(ICHAN(J),DBLK,300)
        IF(II.LT.0) TYPE 450,II
C       write (anything) to last block on new file, to define its
C       length
        II=IWRITW(30,VAL,299,ICHAN(J))
C       file is open
5       A(5,IOP)=1
        RETURN
C       close file
10      TYPE 999
999     FORMAT(1X,'FILOPC - CLOSE')
        CALL ICLOSE(ICHAN(J),II)
        TYPE 1000,ICHAN(J),II
1000    FORMAT(1X,'CLOSING, CHANNEL ',I2,' CODE ',I2)
C       file is closed
        A(5,ICLO)=0
        RETURN
350     FORMAT(1X,'LOOKUP ERROR. RETURN CODE IS ',I3)
450     FORMAT(1X,'IENTER ERROR. RETURN CODE IS ',I3)
        END
```

```
      SUBROUTINE UPDATE(J,IMOD)
C     THIS SUBROUTINE UPDATES CELL PARAMETERS IN GMAT.UNF
      COMMON/PASS/A,B,JT,VAL,QO,Q
      COMMON/RES/JTOP,JMAX
      COMMON/CYCLCT/ICYCLE(16),MCYCLE(16)
      DATA IO,I3/0,3/
      INTEGER A(22,16),B(14,16)
      INTEGER*4 JT(10,16),JTOP,JTOM
      DIMENSION VAL(14,16),QO(2,16),Q(2,16)
C     IMOD=1   AFTER CHANGE
C     IMOD=2   FROM FINISH
C     IMOD=3   AFTER CHECK
C     IMOD=4   FROM MSUPDT
      IF(IMOD.EQ.4) GOTO 20
      IF(IMOD.GT.1) GOTO 5
C
C     after CHANGE, update new cycling specifications
```

```
C
      KPOINT=785+J
      WRITE(12'KPOINT) ICYCLE(J),MCYCLE(J)
      KPOINT=5+49*(J-1)
      WRITE(12'KPOINT) A(1,J),A(2,J)
      KPOINT=6+49*(J-1)
      WRITE(12'KPOINT) A(3,J),I3
      RETURN
C
C     after CHECK or FINISH, update general parameters
C
5     CALL GTIM(JTOP)
      CALL JJCVT(JTOP)
      CALL IDATE(IMONTH,IDAY,IYEAR)
      KPOINT=2+49*(J-1)
      WRITE(12'KPOINT) IMONTH,IDAY
      KPOINT=KPOINT+1
      WRITE(12'KPOINT) IYEAR,I0
      KPOINT=KPOINT+1
      WRITE(12'KPOINT) JTOP
      IF(IMOD.GT.2) GOTO 10
      KPOINT=5+49*(J-1)
C     if called from FINISH, update IN/OUT and ON/OFF flags
      WRITE(12'KPOINT) A(1,J),A(2,J)
10    KPOINT=8+49*(J-1)
      WRITE(12'KPOINT) A(7,J),I0
C     if not single mode, jump
      IF(B(3+A(3,J),J).NE.2.AND.B(3+A(3,J),J).NE.5.AND.
     1B(3+A(3,J),J).NE.8) GOTO 15
C     subtract time from end-of-half-cycle action timer
```

```
      II=JSUB(JT(3,J),JTOP,JTOM)
      KPOINT=28-A(3,J)+49*(J-1)
C     update DTch or DTds, according to mode
      WRITE(12'KPOINT) JTOM
15    KPOINT=47+A(3,J)+49*(J-1)
      WRITE(12'KPOINT) QO(1+A(3,J),J)
      KPOINT=49+49*(J-1)
      WRITE(12'KPOINT) Q(1,J)
      KPOINT=KPOINT+1
      WRITE(12'KPOINT) Q(2,J)
      GOTO 50
C
C     if called from MSUPDT, update all relevant parameters
C
C     if in operation, jump : A array not updated
20    IF(A(1,J).EQ.1) GOTO 30
      KPOINT=9+49*(J-1)
      DO 25 KK=1,7
      WRITE(12'KPOINT) A(7+2*KK,J),A(8+2*KK,J)
25    KPOINT=KPOINT+1
```

```
30      KPOINT=16+49*(J-1)
        READ(12'KPOINT) B(1,J),IDUM
C       if not in operation, or in discharge mode, or not in CVLC
C       charge mode, update B(2,J)
        IF(A(1,J).EQ.0.OR.A(3,J).EQ.0.OR.B(4,J).LT.4.OR.
       1B(4,J).GT.6) IDUM=B(2,J)
        WRITE(12'KPOINT) B(1,J),IDUM
        KPOINT=18+49*(J-1)
        DO 35 KK=2,6
        WRITE(12'KPOINT) B(1+2*KK,J),B(2+2*KK,J)
35      KPOINT=KPOINT+1
        KPOINT=29+49*(J-1)
        WRITE(12'KPOINT) JT(7,J)
        KPOINT=KPOINT+1
        WRITE(12'KPOINT) JT(8,J)
        KPOINT=33+49*(J-1)
        WRITE(12'KPOINT) VAL(1,J)
        KPOINT=KPOINT+1
C       if not in operation, or in discharge mode, or not in CVLC
C       charge mode, update VAL(2,J)
        IF(A(1,J).EQ.0.OR.A(3,J).EQ.0.OR.B(4,J).LT.4.OR.
       1B(4,J).LT.4) WRITE(12'KPOINT) VAL(2,J)
        KPOINT=KPOINT+1
C       update calibration factor only if not in operation
        IF(A(1,J).EQ.0) WRITE(12'KPOINT) VAL(3,J)
        KPOINT=KPOINT+1
        DO 40 KK=1,9
        WRITE(12'KPOINT) VAL(3+KK,J)
40      KPOINT=KPOINT+1
        KPOINT=47+49*(J-1)
```

```
      WRITE(12'KPOINT) Q0(1,J)
C     write to lst and last record of GMAT.UNF; this ensures
C     that the information is actually written on the file
50    READ(12'1) JDUM,IDUM
      WRITE(12'1) JDUM,IDUM
      READ(12'785) QQ
      WRITE(12'785) QQ
      RETURN
      END


      SUBROUTINE FINISH(IND,IALL,BUF)
C     THIS SUBROUTINE TERMINATES THE OPERATION OF 1 (IALL=0) OR
```

```
C       ALL (IALL=1) CELLS
        COMMON/PASS/A,B,JT,VAL,QO,Q
        COMMON/CHANNELS/ICHAN(16)
        COMMON/RES/JTOP,JMAX
        INTEGER A(22,16),B(14,16)
        DIMENSION VAL(14,16),QO(2,16),Q(2,16)
        DIMENSION BUF(128,16)
        INTEGER*4 JT(10,16),JTOP
        TYPE 9300
9300    FORMAT(1X,'FINISH')
        IF(IALL.EQ.1) TYPE 9400
9400    FORMAT(1X,'A DOUBLE ^C WAS TYPED')
C       get time
        CALL GTIM(JTOP)
        CALL JJCVT(JTOP)
C       if all cells are terminated, jump
        IF(IALL.EQ.1) GOTO 1
C       only one pass in coming DO loop
        JSTART=IND
        JEND=IND
        GOTO 2
C       coming DO loop is for all cells
1       JSTART=1
        JEND=JMAX
2       DO 10 J=JSTART,JEND
C       if cell not in operation, jump
        IF(A(1,J).EQ.0) GOTO 10
C       wait for I/O termination on the cell's channel
        I=IWAIT(ICHAN(J))
C       if data file open for cell, skip
```

```
      IF(A(5,J).EQ.1) GOTO 5
C     open file
      CALL FILOPC(J,0)
C     set NSTOP flag in pointer storage area (in BUF)
5     BUF(124,J)=A(6,J)
C     if no data records stored on the present block, NSTOP=300
      IF(A(6,J).EQ.0) BUF(124,J)=300.
C     cell goes out of operation
      A(1,J)=0
C     if not in o-c, terminate control
      IF(A(2,J).EQ.1) CALL CHOOSE(J,3,0,BUF)
C     update parameters in GMAT.UNF
      CALL UPDATE(J,2)
C     write data of present block to the file
      II=IWRITW(256,BUF(1,J),A(7,J),ICHAN(J))
C     close the file
      CALL FILOPC(0,J)
C     if cell not controlling, jump
      IF(A(11,J).GE.J) GOTO 10
```

```
C       set current 0
        CALL PLACE(A(9,J),0.,A(10,J),IDUM)
C       if cell was in pulsed current mode, set Controller in normal
C       mode
        IF(B(3+A(3,J),J).GE.7) CALL EXT(0,A(9,J))
C       disconnect
        CALL SET(A(1,J),0,A(9,J))
10      CONTINUE
C       if only one cell, return
        IF(IALL.EQ.0) RETURN
C       disable ^C^C intercept function
        CALL SCCA
        STOP
        END



        SUBROUTINE RESET(ID)
C       THIS SUBROUTINE TAKES CARE OF THE TRANSITION FORM MIDNIGHT
C       TO THE NEXT DAY: AT THAT MOMENT, THE TIME OF DAY GOES BACK
C       TO 0
        COMMON/PASS/A,B,JT,VAL,Q0,Q
        COMMON/CURTIM/JLAST,JREM
        COMMON/RES/JTOP,JMAX
        COMMON/GDNIT/JMIN
        INTEGER A(22,16),B(14,16)
        INTEGER*2 LINK(4)
        INTEGER*4 JT(10,16),JLAST(16),JREM(16),JTOP,JMID,JTIP,JMIN
```

```
      DIMENSION VAL(14,16),QO(2,16),Q(2,16)
C     JMID = 24 hrs
      CALL JTIME(24,0,0,0,JMID)
      CALL JJCVT(JMID)
C     JTIP = 23:59:50 hrs
      CALL JTIME(23,59,50,0,JTIP)
      CALL JJCVT(JTIP)
C     subtract 24 hrs from all timers
      DO 10 J=1,JMAX
      DO 5 K=1,3
5     II=JSUB(JT(K,J),JMID,JT(K,J))
      II=JSUB(JT(9,J),JMID,JT(9,J))
      II=JSUB(JLAST(J),JMID,JLAST(J))
      II=JSUB(JREM(J),JMID,JREM(J))
10    CONTINUE
      II=JSUB(JMIN,JMID,JMIN)
11    DO 15 K=1,1000
C     get time of day
      CALL GTIM(JTOP)
      CALL JJCVT(JTOP)
C     if time of day < JTIP, done
15    IF(JCMP(JTOP,JTIP).LE.0) GOTO 20
      GOTO 11
20    RETURN
      END
```

```
      SUBROUTINE CRMESS
C     THIS COMPLETION ROUTINE IS THE MESSAGE RECEIVER
      COMMON/MESSAG/MSG,MESCNT,AREA
      INTEGER MSG(59),AREA(4)
C     set flag for message pending
      MESCNT=MESCNT+1
      RETURN
      END


      SUBROUTINE MSBOSS(INITE,BUF)
C     THIS SUBROUTINE IS THE MESSAGE HANDLER
      COMMON/MESSAG/MSG,MESCNT,AREA,AREA2,IMSTOP
      COMMON/CCHOLD/JCYCLE(16),KCYCLE(16),IHOLD
      EXTERNAL CRMESS
      INTEGER MSG(59),AREA(4),RESP(1),AREA2(4,16)
      DIMENSION BUF(128,16)
      IDONE=0
      IF(IHOLD.EQ.1) CALL MSHOLD(IDONE,1)
      IF(IMSTOP.NE.0) CALL MSNOW
C     if no message pending, return
      IF(MESCNT.EQ.0) RETURN
      TYPE 1000
1000  FORMAT(61X,'THERE IS A MESSAGE')
C     call appropriate routine
      IF(MSG(2).EQ.'CH') CALL MSCHNG(IDONE)
      IF(MSG(2).EQ.'DS') CALL MSCHNG(IDONE)
```

```
        IF(MSG(2).EQ.'T1') CALL MSUPDT(IDONE)
        IF(MSG(2).EQ.'ST') CALL MSWHEN(IDONE)
        IF(MSG(2).EQ.'HD') CALL MSWHEN(IDONE)
        IF(MSG(2).EQ.'T2') CALL MSSTOP(IDONE)
        IF(MSG(2).EQ.'T3') CALL MSHOLD(IDONE,0)
        IF(MSG(2).EQ.'RR') CALL MSWHO(IDONE)
        IF(MSG(2).EQ.'T4') CALL MSRUN(IDONE,INITE,BUF)
        IF(IDONE.EQ.1) TYPE 2000,MSG(2)
2000    FORMAT(51X,'MESSAGE-ACTION ',A2,' EXECUTED')
        IF(IDONE.EQ.1) RETURN
C       if no action done, send back response 'BD'
        RESP(1)='BD'
        CALL ISDAT(RESP,1)
C       reset message flag
        MESCNT=MESCNT-1
C       activate message receiver
        II=IRCVDF(MSG,3,AREA,CRMESS)
        TYPE 3000,MSG(2)
3000    FORMAT(31X,'NO MESSSAGE-ACTION EXECUTED. MESSAGE CODE WAS ',A2)
        RETURN
        END


        SUBROUTINE MSWHO(IDONE)
C       THIS SUBROUTINE EXECUTES AT THE 1ST PASS OF THE RUN OPERATION
        COMMON/MESSAG/MSG,MESCNT,AREA,AREA2,IMSTOP
        COMMON/PASS/A,B,JT,VAL,QO,Q
```

```
      COMMON/RES/JTOP,JMAX
      COMMON/PNTRS/NPOINT
      EXTERNAL CRMESS
      INTEGER A(22,16),B(14,16),MSG(59),AREA(4),RESP(12),NPOINT(2),
     1AREA2(4,16)
      INTEGER*4 JT(10,16),JTOM,JTOP
      DIMENSION VAL(14,16),QO(2,16),Q(2,16)
      EQUIVALENCE(JTOM,RESP(5))
C     reset message flag
      MESCNT=MESCNT-1
      MM=MSG(3)
C     if cell # < or = JMAX, jump
      IF(MM.LE.JMAX) GOTO 5
C     too high !
      RESP(1)='TH'
      GOTO 15
C     if cell not in operation, jump
5     IF(A(1,MM).EQ.0) GOTO 10
C     no good !
      RESP(1)='NG'
      GOTO 15
10    RESP(1)='OK'
C     read latest cycling information from GMAT.UNF
      NPOINT(2)=2+49*(MM-1)
      READ(12'NPOINT(2)) RESP(2),RESP(3)
      READ(12'NPOINT(2)) RESP(4),IDUM
      READ(12'NPOINT(2)) JTOM
      NPOINT(2)=5+49*(MM-1)
      READ(12'NPOINT(2)) RESP(7),RESP(8)
      READ(12'NPOINT(2)) IA3,IDUM
```

```
C       if things look too funny for you, jump
        IF(IABS(RESP(7)).GT.1.OR.IABS(RESP(8)).GT.1.OR.IABS(IA3).GT.1)
        1GOTO 15
C       RESP(9) will contain flag for last active mode (0 for ds, 1
C       for ch)
        RESP(9)=IA3*RESP(8)+(1-IA3)*(1-RESP(8))
        NPOINT(2)=8+49*(MM-1)
        READ(12'NPOINT(2)) RESP(10),IDUM
        NPOINT(2)=785+MM
        READ(12'NPOINT(2)) RESP(11),RESP(12)
C       send back response
15      II=ISDAT(RESP,12)
        IDONE=1
C       activate receiver
        II=IRCVDF(MSG,8,AREA,CRMESS)
        RETURN
        END



        SUBROUTINE MSRUN(IDONE,INITE,BUF)
C       THIS SUBROUTINE EXECUTES AT THE 2ND (AND 3RD) PASS OF THE
C       RUN OPERATION
        COMMON/MESSAG/MSG,MESCNT,AREA,AREA2,IMSTOP
        COMMON/PASS/A,B,JT,VAL,QO,Q
        COMMON/PNTRS/NPOINT
        COMMON/INT4/JTWO,JFOUR,JFAC
        COMMON/CHANNELS/ICHAN(16)
```

```
      COMMON/CYCLCT/ICYCLE(16),MCYCLE(16)
      COMMON/CCHOLD/JCYCLE(16),KCYCLE(16),IHOLD
      EXTERNAL CRMESS
      INTEGER A(22,16),B(14,16),MSG(59),AREA(4),RESP(2),NPOINT(2),
     1AREA2(4,16)
      INTEGER*4 JT(10,16),JTWO,JFOUR,JFAC(16)
      DIMENSION VAL(14,16),BUF(128,16),QO(2,16),Q(2,16)
      EQUIVALENCE(NPOINT(1),JPOINT)
      NN1=7
C     reset message flag
      MESCNT=MESCNT-1
      INITE=1
      MM=MSG(3)
C     decode cycling specifications, including the following flags:
C             ISTART: 1 for 'initialize', 2 for 're-start' mode
C             ISCRAT: 1 for new scratch-file, 0 otherwise
C             ISERIE: 1 for 2nd pass of RUN operation
C                     2 for 3rd pass (only if 2 cells in series)
      ISTART=MSG(4)
      ISCRAT=MSG(7)
      ICYCLE(MM)=MSG(5)
      MCYCLE(MM)=MSG(6)
      ISERIE=MSG(8)
C     open FMAT.UNF
      OPEN(UNIT=10,NAME='VOL:FMAT.UNF',TYPE='OLD',ACCESS='DIRECT',
     1RECORDSIZE=1,ERR=999,ASSOCIATEVARIABLE=JPOINT)
      GOTO(5,10) ISTART
999   WRITE(NN1,1000)
1000  FORMAT(1X,'ERROR IN OPENING FMAT.UNF')
C     error !
```

```
      RESP(1)='EF'
      GOTO 110
C     'initialize' mode
5     IPT=1
      IUN=10
      GOTO 15
C     're-start' mode
10    IPT=2
      IUN=12
C     read parameters from FMAT.UNF or GMAT.UNF
15    NPOINT(IPT)=5+49*(MM-1)
      DO 20 I=1,21,2
20    READ(IUN'NPOINT(IPT)) A(I,MM),A(I+1,MM)
      DO 25 I=1,13,2
25    READ(IUN'NPOINT(IPT)) B(I,MM),B(I+1,MM)
      DO 30 I=1,10
30    READ(IUN'NPOINT(IPT)) JT(I,MM)
      DO 35 I=1,14
35    READ(IUN'NPOINT(IPT)) VAL(I,MM)
      DO 40 I=1,2
40    READ(IUN'NPOINT(IPT)) QO(I,MM)
      DO 45 I=1,2
45    READ(IUN'NPOINT(IPT)) Q(I,MM)
C     if 're-start' mode, jump
      IF(ISTART.EQ.2) GOTO 80
      NPOINT(2)=8+49*(MM-1)
C     if no new scratch-file, get number of last block written into
      IF(ISCRAT.EQ.0) READ(12'NPOINT(2)) A(7,MM),IDUM
C     copy all FMAT.UNF parameters into GMAT.UNF
      NPOINT(2)=5+49*(MM-1)
```

```
      DO 50 I=1,21,2
50    WRITE(12'NPOINT(2)) A(I,MM),A(I+1,MM)
      DO 55 I=1,13,2
55    WRITE(12'NPOINT(2)) B(I,MM),B(I+1,MM)
      DO 60 I=1,10
60    WRITE(12'NPOINT(2)) JT(I,MM)
      DO 65 I=1,14
65    WRITE(12'NPOINT(2)) VAL(I,MM)
      DO 70 I=1,2
70    WRITE(12'NPOINT(2)) QO(I,MM)
      DO 75 I=1,2
75    WRITE(12'NPOINT(2)) Q(I,MM)
C     if no new scratch-file, jump
      IF(ISCRAT.EQ.0) GOTO 85
      GOTO 90
C     if no new scratch-file, jump
80    IF(ISCRAT.EQ.0) GOTO 85
C     set block number = 0 in GMAT.UNF
      NPOINT(2)=8+49*(MM-1)
      READ(12'NPOINT(2)) A(7,MM),A(8,MM)
      A(7,MM)=0
      NPOINT(2)=NPOINT(2)-1
      WRITE(12'NPOINT(2)) A(7,MM),A(8,MM)
      GOTO 90
C     open scratch-file
85    CALL FILOPC(MM,0)
      NPOINT(2)=8+49*(MM-1)
      READ(12'NPOINT(2)) A(7,MM),A(8,MM)
C     set INTERR = 1 in the last block's pointer storage area
      II=IREADW(256,BUF(1,MM),A(7,MM),ICHAN(MM))
```

```
      BUF(127,MM)=1
      II=IWRITW(256,BUF(1,MM),A(7,MM),ICHAN(MM))
C     close scratch-file
      CALL FILOPC(0,MM)
C     increment block number and update it in GMAT.UNF
      A(7,MM)=A(7,MM)+1
      NPOINT(2)=NPOINT(2)-1
      WRITE(12'NPOINT(2)) A(7,MM),A(8,MM)
C     initialize pointer and A-hr storage area in BUF
90    DO 95 I=1,8
95    BUF(120+I,MM)=0.
C     compute JFAC (see explanation of TIMEAS routine)
      CALL JJCVT(JT(4,MM))
      CALL CVTTIM(JT(4,MM),IHR,IMIN,ISCND,ITICK)
      CALL JJCVT(JT(4,MM))
      FAC=FLOAT(IMIN)+FLOAT(ISCND)/60.+FLOAT(ITICK)/3600.
      IF(FAC.GT.16.) FAC=16.
      FAC=ALOG(FAC*2.)/ALOG(2.)
      IFAC=FAC
      IF((FAC-IFAC).GT.(.5)) IFAC=IFAC+1
      FAC=2.**(FLOAT(IFAC)/2)
      II=JAFIX(FAC,JFAC(MM))
C     if cycle-update mode is the same as starting mode, then the
C     cycle # transferred from WATCHDOG is 1 too high: it will
C     automatically get incremented in subroutine INPUT;
C     decrement it now
      IF(MCYCLE(MM).EQ.A(3,MM)) ICYCLE(MM)=ICYCLE(MM)-1
C     no HOLD request pending for the cell
      KCYCLE(MM)=-1
C     close FMAT.UNF
```

```
      CLOSE(UNIT=10)
C     if no cell in series, jump
      IF(A(11,MM).EQ.0) GOTO 105
C     if 3rd pass of RUN operation, jump
      IF(ISERIE.EQ.2) GOTO 100
C     there is a cell in series !..
      RESP(1)='SC'
C     ...and that's its number
      RESP(2)=A(11,MM)
C     the 1st cell must wait
      A(1,MM)=0
C     send response
      II=ISDAT(RESP,2)
      IDONE=1
C     activate receiver
      II=IRCVDF(MSG,8,AREA,CRMESS)
      RETURN
C     set the ignition for the companion cell
100   A(1,A(11,MM))=1
C     it starts from an o-c condition
      A(2,A(11,MM))=0
      NPOINT(2)=5+49*(A(11,MM)-1)
C     quick ! update these parameters
      WRITE(12'NPOINT(2)) A(1,A(11,MM)),A(2,A(11,MM))
C     set the ignition
105   A(1,MM)=1
C     start from an o-c condition
      A(2,MM)=0
      NPOINT(2)=5+49*(MM-1)
C     n'oubliez pas to update these parameters
```

```
      WRITE(12'NPOINT(2)) A(1,MM),A(2,MM)
C     here we go....
      RESP(1)='OK'
C     send response
110   II=ISDAT(RESP,1)
      IDONE=1
C     activate receiver
      II=IRCVDF(MSG,3,AREA,CRMESS)
      RETURN
      END



      SUBROUTINE MSCHNG(IDONE)
C     THIS SUBROUTINE EXECUTES AT THE 1ST PASS OF THE CHANGE OR
C     DISPLAY OPERATION
      COMMON/MESSAG/MSG,MESCNT,AREA,AREA2,IMSTOP
      COMMON/PASS/A,B,JT,VAL,QO,Q
      COMMON/PNTRS/NPOINT
      COMMON/CYCLCT/ICYCLE(16),MCYCLE(16)
      EXTERNAL CRMESS
      INTEGER MSG(59),AREA(4),RESP(80),A(22,16),B(14,16),AREA2(4,16),
     1NPOINT(2)
      INTEGER*4 JT(10,16),JJT(5)
      DIMENSION VAL(14,16),QO(2,16),Q(2,16),VVAL(14),QQO(2)
      EQUIVALENCE (RESP(38),JJT(1))
      EQUIVALENCE (RESP(48),VVAL(1))
      EQUIVALENCE (RESP(76),QQO(1))
```

```
      MESCNT=MESCNT-1
      MM=MSG(3)
      RESP(1)='OK'
C     if cell in operation, jump
      IF(A(1,MM).EQ.1) GOTO 30
C     read parameters from GMAT.UNF
      NPOINT(2)=5+49*(MM-1)
      DO 5 I=1,21,2
5     READ(12'NPOINT(2)) A(I,MM),A(I+1,MM)
C     do not allow cell to go in operation !
      IF(A(1,MM).EQ.1) A(1,MM)=0
      DO 10 I=1,13,2
10    READ(12'NPOINT(2)) B(I,MM),B(I+1,MM)
      DO 15 I=1,10
15    READ(12'NPOINT(2)) JT(I,MM)
      DO 20 I=1,14
20    READ(12'NPOINT(2)) VAL(I,MM)
      DO 25 I=1,2
25    READ(12'NPOINT(2)) QO(I,MM)
      NPOINT(2)=785+MM
      READ(12'NPOINT(2)) ICYCLE(MM),MCYCLE(MM)
C     encode response
30    DO 35 K=1,22
35    RESP(1+K)=A(K,MM)
      DO 40 K=1,14
40    RESP(23+K)=B(K,MM)
      DO 45 K=1,5
45    II=JMOV(JT(3+K,MM),JJT(K))
      DO 50 K=1,14
50    VVAL(K)=VAL(K,MM)
```

```
      DO 55 K=1,2
55    QQO(K)=QO(K,MM)
      RESP(80)=ICYCLE(MM)
C     send response
      II=ISDAT(RESP,80)
      IDONE=1
C     activate message receiver
      II=IRCVDF(MSG,59,AREA,CRMESS)
      RETURN
      END



      SUBROUTINE MSUPDT(IDONE)
C     THIS SUBROUTINE EXECUTES AT THE 2ND PASS OF THE CHANGE OPERATION
      COMMON/MESSAG/MSG,MESCNT,AREA,AREA2,IMSTOP
      COMMON/INT4/JTWO,JFOUR,JFAC
      COMMON/PASS/A,B,JT,VAL,QO,Q
      COMMON/CYCLCT/ICYCLE(16),MCYCLE(16)
      COMMON/TIMTIM/JCHECK,JSET,JCVLC
      EXTERNAL CRMESS
      INTEGER MSG(59),MSG1(59),AREA(4),RESP(1),A(22,16),B(14,16),
     1AREA2(4,16)
      INTEGER*4 JT(10,16),JT4,JT7,JT8,JTWO,JFOUR,JFAC(16),JTEST,
     1JCHECK,JSET,JCVLC,JFF
      DIMENSION VAL(14,16),VVAL(12),QO(2,16),Q(2,16)
      EQUIVALENCE (MSG1(26),JT4)
      EQUIVALENCE (MSG1(28),JT7)
```

```
      EQUIVALENCE (MSG1(30),JT8)
      EQUIVALENCE (MSG1(32),JFF)
      EQUIVALENCE (MSG1(34),VVAL(1))
      EQUIVALENCE (MSG1(58),QQ0)
C     reset message flag
      MESCNT=MESCNT-1
C     copy message
      DO 5 K=1,59
5     MSG1(K)=MSG(K)
      MM=MSG1(3)
C     if cell in operation, do not update A array !
      IF(A(1,MM).EQ.1) GOTO 15
C
C           update of A array
C
      DO 10 K=1,14
10    A(8+K,MM)=MSG1(3+K)
C
C           update of B array
C
15    B(1,MM)=MSG1(18)
C     if not in operation, or in discharge mode, or not in CVLC
C     charge mode, update B(2,J)
      IF(A(1,MM).EQ.0.OR.A(3,MM).EQ.0.OR.B(4,MM).LT.4.OR.
     1B(4,MM).GT.6) B(2,MM)=MSG1(19)
      DO 20 K=1,6
20    B(4+K,MM)=MSG1(19+K)
C
C           update of JT array
C
```

```
C       if JFAC (and DTstore) was not modified, jump
        IF(MSG1(32).EQ.0.AND.MSG1(33).EQ.0) GOTO 30
C       update JFAC
        II=JMOV(JFF,JFAC(MM))
        KPOINT=26+49*(MM-1)
C       update DTstore in GMAT.UNF
        WRITE(12'KPOINT) JT4
C       what is the present value of DTinp ? (JSET = 1 min.)
        II=JDIV(JSET,JT(10,J),JTEST)
C       if DTinp is more than 30 seconds, jump
        IF(JCMP(JTEST,JTWO).LT.0) GOTO 25
C       if it is less or equal to 30 seconds, divide DTstore by JFAC
        II=JDIV(JT4,JFAC(MM),JT4)
C       if it is less than 30 seconds, divide DTstore by JFAC a 2nd
C       time
        IF(JCMP(JTEST,JFOUR).GE.0) II=JDIV(JT4,JFAC(MM),JT4)
25      II=JMOV(JT4,JT(4,MM))
30      II=JMOV(JT7,JT(7,MM))
        II=JMOV(JT8,JT(8,MM))
C
C               update of VAL array
C
        VAL(1,MM)=VVAL(1)
C       if not in operation, or in discharge mode, or not in CVLC
C       charge mode, update VAL(2,J)
        IF(A(1,MM).EQ.0.OR.A(3,MM).EQ.0.OR.B(4,MM).LT.4.OR.
       1B(4,MM).GT.6) VAL(2,MM)=VVAL(2)
C       if not in operation, update calibration factor
        IF(A(1,MM).EQ.0) VAL(3,MM)=VVAL(3)
40      DO 45 K=1,9
```

```
45      VAL(3+K,MM)=VVAL(3+K)
C
C               update of Q and Q0 arrays
C
        Q0(1,MM)=QQ0
C       update all these parameters in GMAT.UNF
        CALL UPDATE(MM,4)
        RESP(1)='OK'
C       send response
        II=ISDAT(RESP,1)
        IDONE=1
C       activate message receiver
        II=IRCVDF(MSG,3,AREA,CRMESS)
        RETURN
        END



        SUBROUTINE MSWHEN(IDONE)
C       THIS SUBROUTINE EXECUTES AT THE 1ST PASS OF THE HOLD OR STOP
C       REQUEST
        COMMON/MESSAG/MSG,MESCNT,AREA,AREA2,IMSTOP
        COMMON/PASS/A,B,JT,VAL,Q0,Q
        COMMON/CYCLCT/ICYCLE(16),MCYCLE(16)
        COMMON/CCHOLD/JCYCLE(16),KCYCLE(16),IHOLD
        EXTERNAL CRMESS
        INTEGER MSG(59),AREA(4),RESP(10),A(22,16),B(14,16),AREA2(4,16)
        INTEGER*4 JT(10,16),JTLEFT
```

```
      DIMENSION VAL(14,16),QO(2,16),Q(2,16)
      EQUIVALENCE (RESP(2),JTLEFT)
      EQUIVALENCE (RESP(5),QQO)
      EQUIVALENCE (RESP(7),QQ)
C     reset message flag
      MESCNT=MESCNT-1
      MM=MSG(3)
C     if cell in operation, jump
      IF(A(1,MM).NE.0) GOTO 1
C     not connected !
      RESP(1)='NC'
C     send response
      II=ISDAT(RESP,1)
      GOTO 10
C     cancel eventual marktime for the cell
1     II=ICMKT(MM,JTLEFT)
C     if there was no marktime, jump
      IF(II.NE.0) GOTO 5
      CALL JJCVT(JTLEFT)
C     no good !
      RESP(1)='NG'
C     send back response (with the time that was left for the
C     marktime when it got cancelled)
      II=ISDAT(RESP,3)
      GOTO 10
C     response will contain cycling information
5     RESP(1)='OK'
      RESP(2)=ICYCLE(MM)
      RESP(3)=A(2,MM)
      RESP(4)=A(3,MM)
```

```
      QQ0=Q0(1+A(3,MM),MM)*A(10,MM)/5.
      QQ=Q(1+A(3,MM),MM)*A(10,MM)/5.
      RESP(9)=JCYCLE(MM)
      RESP(10)=KCYCLE(MM)
C     send response
      II=ISDAT(RESP,10)
10    IDONE=1
C     activate message receiver
      II=IRCVDF(MSG,7,AREA,CRMESS)
C     this is a mistery to me ! -- do not remove this statement, or
C     the program will occasionally bom
      MESCNT=0
      RETURN
      END



      SUBROUTINE MSSTOP(IDONE)
C     THIS SUBROUTINE EXECUTES AT THE 2ND PASS OF THE STOP OPERATION
      COMMON/MESSAG/MSG,MESCNT,AREA,AREA2,IMSTOP
      EXTERNAL CRSTOP,CRMESS
      INTEGER MSG(59),AREA(4),RESP(1),AREA2(4,16)
C     reset message flag
      MESCNT=MESCNT-1
C     activate timer request for the cell
      II=ITIMER(MSG(4),MSG(5),MSG(6),MSG(7),AREA2(1,MSG(3)),MSG(3),
     1CRSTOP)
      RESP(1)='OK'
```

```
C       send response
        II=ISDAT(RESP,1)
        IDONE=1
C       activate receiver
        II=IRCVDF(MSG,3,AREA,CRMESS)
        RETURN
        END


        SUBROUTINE CRSTOP(MM)
C       THIS COMPLETION ROUTINE IS CALLED WHEN A STOP-TIMER REQUEST
C       EXPIRES
        COMMON/MESSAG/MSG,MESCNT,AREA,AREA2,IMSTOP
        INTEGER MM,MSG(59),AREA(4),AREA2(4,16)
C       set flag for timer request expired
        IMSTOP=MM
        RETURN
        END


        SUBROUTINE MSHOLD(IDONE,MODE)
C       THIS SUBROUTINE EXECUTES AT THE 2ND PASS OF THE HOLD OPERATION
C       (IMODE=0), OR WHEN A CELL GOES IN OPEN-CIRCUIT (IMODE=1)
        COMMON/MESSAG/MSG,MESCNT,AREA,AREA2,IMSTOP
```

```
      COMMON/PASS/A,B,JT,VAL,QO,Q
      COMMON/CYCLCT/ICYCLE(16),MCYCLE(16)
      COMMON/CCHOLD/JCYCLE(16),KCYCLE(16),IHOLD
      EXTERNAL CRMESS
      INTEGER MSG(59),AREA(4),RESP(1),A(22,16),B(14,16),AREA2(4,16)
      INTEGER*4 JT(10,16)
      DIMENSION VAL(14,16),QO(2,16),Q(2,16)
      GOTO(5,20) MODE+1
5     MM=MSG(3)
C     reset message flag
C
C     now we are going to check whether or not the request is valid:
C     this depends on the cycle and half-cycle at which the cell
C     is operating, and the cycle and half-cycle at which the cell
C     is requested to stop; the latter information is contained
C     in MSG(4) and MSG(5)
C             MSG(4) contains the cycle number at which the cell
C                    must stop
C             MSG(5) contains the mode (0 for ds, 1 for ch) of
C                    the half-cycle (for that cycle number)
C                    after which the cell must stop
      MESCNT=MESCNT-1
C     if you are not yet at the cycle where you are asked to
C     stop, jump: the request is valid
      IF(MSG(4).GT.ICYCLE(MM)) GOTO 10
C     IMODE is the mode (0 for ds, 1 for ch) of the present or last
C     half-cycle (last if already in o-c)
      IMODE=A(3,MM)*A(2,MM)+(1-A(3,MM))*(1-A(2,MM))
C     if the present and the requested cycle number are the same
C     then:
```

```
C               case of a cell in o-c:
C                       if the mode of the requested half-cycle is
C                       the same as that of the last half-cycle,
C                       valid; jump
C               case of a cell on ch or ds:
C                       if the mode of the requested half-cycle is
C                       the same as that of the present half-cycle,
C                       valid; jump
C               case of cell in o-c, ch or ds:
C                       if the mode of the requested half-cycle is not
C                       the same as the mode at which a cycle number
C                       gets incremented, valid; jump
        IF(MSG(4).EQ.ICYCLE(MM).AND.((MCYCLE(MM).NE.MSG(5)).OR.
        1(IMODE.EQ.MSG(5)))) GOTO 10
C       no good !
        RESP(1)='NG'
        GOTO 15
C       the request is vaild (OK):
C       define cycle and half-cycle at which the cell must stop
10      JCYCLE(MM)=MSG(4)
        KCYCLE(MM)=MSG(5)
        RESP(1)='OK'
C       send response
15      II=ISDAT(RESP,1)
        IDONE=1
C       activate message receiver
        II=IRCVDF(MSG,3,AREA,CRMESS)
C       if cell in o-c and request is valid, jump: maybe the cell
C       has to be stopped right now !
        IF(A(2,MM).EQ.0.AND.RESP(1).EQ.'OK') GOTO 20
```

```
      RETURN
C     reset flag that was set when 'a' cell went in o-c
20    IHOLD=0
C     DO loop for all cells
      DO 25 K=1,16
C     if no request pending for the cell, or cell not in o-c, jump
      IF(KCYCLE(K).EQ.-1.OR.A(2,K).NE.0) GOTO 25
C     if mode of last half-cycle (1-A(3,K)) not equal to mode of
C     requested half cycle, or cycle number not equal number of
C     requested cycle, jump
      IF((1-A(3,K)).NE.KCYCLE(K).OR.JCYCLE(K).NE.ICYCLE(K)) GOTO 25
C     set termination flag
      A(1,K)=2
C     if there is a companion, set its termination flag
      IF(A(11,K).NE.0) A(1,A(11,K))=2
C     no more HOLD request pending for the cell
      KCYCLE(K)=-1
25    CONTINUE
      RETURN
      END


      SUBROUTINE MSNOW
C     THIS SUBROUTINE IS CALLED BY THE MESSAGE HANDLER:
C     THIS HAPPENS WHEN THE HANDLER IS NOTIFIED (IMSTOP FLAG SET)
C     THAT A STOP-TIMER REQUEST JUST EXPIRED
      COMMON/PASS/A,B,JT,VAL,QO,Q
```

```
      COMMON/MESSAG/MSG,MESCNT,AREA,AREA2,IMSTOP
      INTEGER MM,A(22,16),B(14,16),MSG(59),AREA(4),AREA2(4,16)
      INTEGER*4 JT(10,16),JTOP
      DIMENSION VAL(14,16),QO(2,16),Q(2,16)
C     get cell number from IMSTOP flag
      MM=IMSTOP
C     get time
      CALL GTIM(JTOP)
      CALL JJCVT(JTOP)
C     reset IMSTOP
      IMSTOP=0
C     if cell not in operation, set termination flag
      IF(A(1,MM).NE.0) A(1,MM)=2
C     if there is a companion cell, set its termination flag
      IF(A(11,MM).NE.0) A(1,A(11,MM))=2
      RETURN
      END



      FUNCTION RDLOGG(ICHAN,IRAN,IREAD,ITIME,JCEL)
C     THIS FUNCTION DECIDES WHETHER RLOGG OR RMLOGG SHOULD BE
C     USED FOR READING AN A/D CHANNEL
      COMMON/PASS/A,B,JT,VAL,QO,Q
      INTEGER A(22,16),B(14,16)
      INTEGER*4 JT(10,16)
      DIMENSION VAL(14,16),QO(2,16),Q(2,16)
C     if control mode > or = 7 (pulsed mode), jump
```

```
      IF(B(3+A(3,JCEL),JCEL).GE.7) GOTO 10
C     regular A/D function: result of RDLOGG is result of RLOGG
5     RDLOGG=RLOGG(ICHAN,IRAN,IREAD,ITIME)
      RETURN
C
C             pulsed current mode
C
C     if current measurement, jump back to regular A/D function
10    IF(ICHAN.EQ.A(19,JCEL)) GOTO 5
C     if A/D range < 100, jump
      IF(IRAN.LT.100) GOTO 15
C     this is a measurement at the ON-side of the pulse: track
C     the pulse and find the maximum value
      MAXMIN=1
C     actual A/D range is given one - 100
      JRAN=IRAN-100
      GOTO 20
C     this is a measurement at the OFF-side of the pulse: track
C     the pulse and find the minimum value
15    MAXMIN=0
C     actual A/D range is given one
      JRAN=IRAN
C     measure for 100 msec.
20    IPER=1000
C     result of RDLOGG is result of RMLOGG
      RDLOGG=RMLOGG(ICHAN,JRAN,IREAD,ITIME,MAXMIN,IPER)
      RETURN
      END
```

REFERENCES

1. RT-11 Documentation, Version 4, Volume 3A, REF, p.1-22, pp.3-34 to 3-36 and 3-45 to 3-48, March 1980. Ditigal Equipment Corporation, Maynard, Mass.

2. RT-11 Documentation Directory, Version 4, Vol. 4, Fortran Language Reference Manual, Jan. 1980. Digital Equipment Corporation, Maynard, Massachusetts.

APPENDIX E

THE WATCHDOG PROGRAM - ORGANIZING CYCLING REGIMES

E.1 CYCLING REGIMES

The WATCHDOG program is written in FORTRAN IV programming language for LSI-11/23 computers.(1) A cycling regime is a combination of a charge and a discharge mode. Charge modes are :

1. constant current
2. constant current - limited voltage (the current is kept constant up to the point where a voltage limit is reached, then the voltage is kept constant)
3. pulsed current

Discharge modes are:

1. constant current
2. constant power
3. pulsed current

Each of these modes has 3 variations:

1. Regular cycling: a cell is continuously charged and discharged (a regular charge and discharge mode must be combined). A charge or discharge can be terminated when a limit on a measured variable is exceeded (e.g. cutoff voltage) or a required number of A-hrs have been passed in that mode. An excess factor is provided, that is multiplied with the discharged A-hrs, so as to define the A-hrs to pass on charge (this accounts for the inefficiency of the cell on charge).

2. One half-cycle: this must be combined with a mode 0 (i.e no mode) for the other half-cycle. A cell is charged or discharged, with the termination procedures described above. An additional feature is the possibility of entering a time limit for the half-cycle, rather than an A-hr limit. When the half-cycle is terminated, the cell stays in open-circuit, taking data for a specified amount of time, then its operation is terminated.

3. Regular cycling for 2 cells in series: two such modes must be combined for charge and dicharge. Both cells under test must have these modes specified. Even though the cycling specifications for each cell are entered separately, it is essential that they be well matched, otherwise the results are unpredictable. In any other aspect, this mode and the first one perform identical functions.

E.2 FUNCTIONS OF THE WATCHDOG PROGRAM

The WATCHDOG program is run in conjunction with the REAL-TIME program. It is the main interface between the user and the electrochemical cell test system. It is through this program that the user can control his experiments: without the WATCHDOG program, the REAL-TIME program is virtually useless.

The watchdog program uses the cell parameters (described in Section D.1) and the files FMAT.UNF and GMAT.UNF (described in Section D.2). It communicates with the REAL-TIME program through a set of messages, with a handling procedure that is described in Section D.5. Subroutines of

the WATCHDOG program are matched with subroutines of the REAL-TIME program: therefore, a view of the routines at both ends is necessary in order to understand the message exchanges.

The WATCHDOG program has the following functions:

IN for INITIALIZE

EX for EXAMINE

DS for DISPLAY

CH for CHANGE

HD for HOLD

ST for STOP

RR for RUN (RR1 and RR2)

E.2.1 INITIALIZE

This function initializes all parameters that characterize the operation of a cell, and stores them in the file FMAT.UNF. There is no communication between the WATCHDOG and the REAL-TIME program: the use of the function is valid regardless of whether the REAL-TIME program runs or not.

E.2.2 EXAMINE

This function displays all the parameters, for a cell, as they were last entered in FMAT.UNF, by the INITIALIZE function. There is no communication between the WATCHDOG and REAL-TIME programs: the use of the function is valid regardless of whether the REAL-TIME program runs or not.

E.2.3 DISPLAY

This function displays all the parameters that describe how a cell is operating at that precise moment. If a cell with a certain number never was in operation, DISPLAY gives inconsistent results.

When the REAL-TIME program is not running (and the cell is, therefore, not in operation), DISPLAY reads the parameters straight from the file GMAT.UNF. When the REAL-TIME program is running, the parameters are sent to the WATCHDOG program by use of a message. In that case, there are two possibilities:

1. If the cell is not in operation, the REAL-TIME program reads the parameters from GMAT.UNF and puts them in the message.

2. If the cell is in operation, the REAL-TIME program sends the contents of the parameter arrays (A,B,JT,VAL,Q,Q0 - see Sec. D.1) at that precise moment. An example of what DISPLAY outputs, in this case, is shown in Table 1.

E.2.4 CHANGE

This function changes selective cell parameters: this can be done while the cell in question is running, and it is, therefore, a way to interactively modify the experiment. In Table 1, only the parameters referenced with a number can be changed: for example, the control modes cannot be changed. This is because they would require too many other changes to be made, in order to keep the cycling conditions meaningful: it is a protection from user errors. When the cell is in operation, some other parameters will not be modified. For example, the current

controller cannot be changed unless the cell is put out of operation and its connections physically modified. A list of parameters that are not modified when the cell is running is given in the description of the MSUPDT subroutine, sec. D.7.29.

When the REAL-TIME program is not running (and the cell is therefore not in operation), CHANGE modifies parameters straight in GMAT.UNF. When the REAL-TIME program is running, the parameters are sent to it in a message. In that case, there are 2 possiblities:

1. if the cell is not in operation, the REAL-TIME program writes the parameters to the file GMAT.UNF.

2. If the cell is in operation, the parameters are read into the parameter arrays, and also written out to GMAT.UNF.

E.2.5 HOLD

This function activates a request to stop a cell after a half-cycle to come. The use of the function is not valid when the REAL-TIME program is not running, or when the cell is not in operation.

HOLD uses subroutine HOLD.

E.2.6 STOP

This function activates a request to stop a cell after a specified amount of time. The use of the function is not valid when the REAL-TIME program is not running, or when the cell is not in operation.

E.2.7 RUN

This function puts a cell in operation. There are 2 modes, RR1 and RR2 (see also sec. D.2).

1. RR1 is the initialize mode. It starts a cell with the parameters that were entered in FMAT.UNF, with the INITIALIZE function, and does not otherwise refer to any previous "history" of the cell. It is up to the user to take that into account when initializing the parameters. He/she has the option of starting a new scratch-data-file, or adding data to the existing one for that cell number, and he/she can also decide at which cycle number the cell starts operating. New cells must always be started in this mode (at cycle 1).

2. RR2 is the re-start mode. It starts a cell with the parameters that are stored in GMAT.UNF. These parameters describe what the cell was doing at the moment it was put out of operation: they reflect its "cycling history". Here, too, the user can choose to start a new scratch-data-file or add data to the existing one, and he can specify the cycle number.

The use of the function is not valid when the REAL-TIME program is not running, or when the cell is already in operation.

E.3 SOME USEFUL HINTS

This section gives a list of useful hints for using the real-time program.

1. The following can happen when a controller is switched in the local mode during a cell's operation:

    1. If the cell is charging or discharging, the current can no longer be controlled by the computer: the control routine sets a termination flag. Possibly, the error can be intercepted first by the hardware check routine. In both cases, the cell is put out of operation.

    2. If the cell is in open-circuit, it is not allowed to start a charge or discharge. The open-circuit period is extended (with the open-circuit time interval), until the controller is set back in the computer mode. The cell stays in operation (data continue to be taken).

2. Only single-charge or single-discharge modes accept time limits instead of A-hr limits. Moreover, they do not put a cell out of operation immediately after the half-cycle: it stays in open-circuit, and data are taken for a specified amount of time. Then, the cell goes out of operation.

3. For the operation of two cells in series, the program does not check whether or not the parameters entered are consistent. The EXAMINE and DISPLAY functions are provided for that use.

4. A half-cycle can be interrupted with a STOP request. The user can re-start it where it left off, but he can also continue the half-cycle with initialized parameters. For example, it is perfectly legal to discharge a cell at constant power to 1 V cutoff voltage, then interrupt it, and continue the same

discharge with new parameters, at constant current, down to 0 V cutoff voltage. However, as in this case, when the cell is run in the "initialize" mode, the A-hrs start counting back at zero: the previous A-hr count not used.

5. A test on 2 cells in series can be interrupted during a half-cycle or at open-circuit, and continued for only one of the cells. The parameters must be re-initialized, to specify that the cell is cycled on its own; if a half-cycle was interrupted, the A-hrs start counting back at zero.

6. During an interruption in a cell's operation, a user can change the current controller it runs on, by using the CHANGE function. However, to do it in that way (rather than with the INITIALIZE function) is only allowed if the range of the controller remains the same (2 A or 10 A).

7. When a cell is in open-circuit, a change in its open-circuit time only takes effect at the next open-circuit.

8. When a cell is in open-circuit, a HOLD request for the previous half-cycle is valid.

9. In order to comply with the restrictions of the data-reducing program, the number of A/D channels should not be changed during the course of a cycle (e.g between charge and discharge). A change between two cycles is allowed.

10. The largest block number used in a scratch-file should never exceed 256.

11. As described in Section E.2.7, when a user places a cell in operation with the RUN function, he/she can choose to start a new scratch-file or add data to an old one.

12. In message exchange procedures (and related WATCHDOG functions), no action is taken until the message "TRANSFER COMPLETED" appears on the screen.

13. A message procedure can be aborted at various points: this has been provided for in the code. However, it can also be aborted by interrupting the WATCHDOG program with a ^C^C.

E.4 SUBROUTINES OF THE WATCHDOG PROGRAM

E.4.1 WATCHD

This is the main program, which types the list of functions on the CRT. The subroutines it calls are located in the extended memory.

E.4.2 ASK

This subroutine is called by MODIF and MATIN. According to the value of the input parameter ITYPE, it types one out of 27 statements on the CRT.

E.4.3 MATIN

MATIN is used by the INITIALIZE function: it asks the user for cell cycling information, and stores it in FMAT.UNF. MATIN checks whether the control modes entered for charge and discharge are consistent with each other, but does not check any other input. The EXAMINE function must be used for that purpose.

E.4.4 MODIF

Depending on the value of the input parameter ICHANG, MODIF is used by the EXAMINE function (-1), the DISPLAY function (0) or the CHANGE function (1).

The first part of the routine is executed for the three functions: the cell parameters are read from FMAT.UNF, GMAT.UNF, or transferred from the REAL-TIME program (see Sections E.2.2 to E.2.4). In the latter case, MODIF communicates with subroutine MSCHNG (Section D.7.28) of the REAL-TIME program. The second part of MODIF is only executed for the CHANGE function: the user can change the value of some cell parameters. A list of parameters that cannot be modified during cell operation is given in Section D.7.29. This part of MODIF communicates with subroutine MSUPDT (Section D.7.29) of the REAL-TIME program.

E.4.5 HOLD

Depending on the value of the input parameter IHOLD, this subroutine is used by the function STOP (0) or HOLD (1). The first part of HOLD is executed for the two functions: it communicates with subroutine MSWHEN (Section D.7.30) of the REAL-TIME program.

The second part of HOLD is executed for the STOP function, and communicates with subroutine MSSTOP (Section D.7.31) of the REAL-TIME program.

The third part is executed for the HOLD function, and communicates with subroutine MSHOLD (Section D.7.33) of the REAL-TIME program.

E.4.6 RUN

This subroutine is used by the RUN function. The first part of RUN communicates with subroutine MSWHO (Section D.7.26) of the REAL-TIME program.

The second part is executed for the "re-start" mode, and the third for the 'initialize' mode. The last part communicates with subroutine MSRUN (Section D.7.27) of the REAL-TIME program.

T A B L E 1

Output of the DISPLAY and the CHANGE(*) Functions, for Cell 10

```
CELL 10 IS RUNNING, CYCLE 44
PRESENT MODE IS CHARGE

LAST BLOCK ON SCRATCH-FILE THAT HAS BEEN WRITTEN TO IS BLOCK NUMBER 118

CURRENT CONTROLLER 2.  FULL-SCALE CURRENT IS 2 A.
     WITH CALIBRATION FACTOR  1.00100                              #  1

4 A/D CHANNELS ACTIVE                                              #  2
   CHANNEL 1 : A/D CHANNEL 11
               RANGE NUMBER 2
   CHANNEL 2 : A/D CHANNEL 12
               RANGE NUMBER 2
   CHANNEL 3 : A/D CHANNEL 13
               RANGE NUMBER 2
   CHANNEL 4 : A/D CHANNEL 14
               RANGE NUMBER 2
   CHANNEL 5 : A/D CHANNEL  0
               RANGE NUMBER 0

ON DISCHARGE:                                                      # 21
   CHANNEL WATCHED FOR REGULATION OF DATA-TAKIING RATE IS 1
   INTERVAL THAT REGULATES THE RATE IS 0.01000
       (IN APPROPRIATE UNITS)

ON CHARGE:                                                         # 22
   CHANNEL WATCHED FOR REGULATION OF DATA-TAKIING RATE IS 1
   INTERVAL THAT REGULATES THE RATE IS 0.01000
       (IN APPROPRIATE UNITS)

DISCHARGE MODE IS 1

CHARGE MODE IS    1

LOWER LIMIT 1 ON DISCHARGE : CHANNEL  1                            # 23
                             VALUE    1.10000

UPPER LIMIT 4 ON    CHARGE : CHANNEL  1                            # 24
                             VALUE    2.15000

DT-MEASUREMENT           0 : 15 : 0 : 0                            # 31

DT-OPEN-CIRCUIT          0 : 10 : 0 : 0                            # 32

DT-CONTROL               0 :  0 : 15: 0                            # 33

CHARGE EXCESS FACTOR     1.04000                                   # 41

CURRENT ON DISCHARGE     0.54000 A                                 # 42

CURRENT ON    CHARGE     0.22500 A                                 # 43

MAXIMUM A-HRS TO BE PASSED ON DISCHARGE  1.35000                   # 51

MAXIMUM A-HRS TO BE PASSED ON    CHARGE  1.40513
```

(*) For the CHANGE function, only the parameters referenced with a member can be modified.

```
      PROGRAM WATCHD
      CALL ASSIGN(6,'LP:/C',5)
      NN1=7
1     WRITE(NN1,1000)
1000  FORMAT(1X,'TYPE  IN  FOR INITIALIZATION',/,1X,T8,'EX  FOR ',
     1'EXAMINE',/,1X,T8,'DS  FOR DISPLAY',/,1X,T8,'CH  FOR CHANGE',
     2/,1X,T8,'HD  FOR HOLD',/,1X,T8,'ST  FOR STOP',/,1X,T8,'RR  ',
     3'FOR RUN')
      READ(5,1100) IMODE
1100  FORMAT(A2)
      IF(IMODE.EQ.'IN') CALL MATIN
      IF(IMODE.EQ.'EX') CALL MODIF(-1)
      IF(IMODE.EQ.'DS') CALL MODIF(0)
      IF(IMODE.EQ.'CH') CALL MODIF(1)
      IF(IMODE.EQ.'HD') CALL HOLD(1)
      IF(IMODE.EQ.'ST') CALL HOLD(0)
      IF(IMODE.EQ.'RR') CALL RUN
      GOTO 1
      END
1
      SUBROUTINE ASK(ITYPE,IND)
      NN1=7
      GOTO (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,
     123,24,25,26,27) ITYPE
```

```
1       WRITE(NN1,1400) IND
1400    FORMAT(1X,'CURRENT CONTROLLER NUMBER ASSOCIATED WITH ',
       1'CELL ',I2,'?')
        RETURN
2       WRITE(NN1,1500)
1500    FORMAT(1X,'GIVE CALIBRATION VALUE FOR THIS CURRENT',
       1' CONTROLLER :(REAL)')
        RETURN
3       WRITE(NN1,1600)
1600    FORMAT(1X,'VALUE(IN A) OF FULL-SCALE CURRENT FOR 50 MV ',
       1'SHUNT :(INTEGER)')
        RETURN
4       WRITE(NN1,1700)
1700    FORMAT(1X,'THERE ARE 5 MEASURING CHANNELS AVAILABLE.',/,1X,
       1'INDICATE NUMBER OF ACTIVE CHANNELS')
        RETURN
```

```
5       WRITE(NN1,1800)
1800    FORMAT(1X,'FOR EACH CHANNEL, GIVE THE CORRESPONDING A/D CHA',
       1'NNEL NUMBER AND A/D RANGE ',/,1X,'CHANNEL 1',
       2' IS CELL VOLTAGE',/,9X,'2',4X,'POSITIVE-REFERENCE',/,9X,
       3'3',4X,'REFERENCE-NEGATIVE',/,9X,'4',4X,'CURRENT',
       4/,9X,'5',4X,'TEMPERATURE',//,1X,'RANGE 0 IS',
       5' SKIP',/,7X,'1',4X,'(+/-)5V',/,7X,'2',4X,'(+/-)2.5V',/,7X,'4',
       64X,'(+/-)1.25V',/,7X,'8',4X,'(+/-).625V',
       7//,1X,'IF CHANNEL NOT ACTIVE, WRITE 0,0')
       RETURN
6       WRITE(NN1,1900) IND
1900    FORMAT(1X,'CHANNEL ',I2,' :A/D CHANNEL NUMBER,RANGE')
       RETURN
7       WRITE(NN1,2000)
2000    FORMAT(1X,'AVERAGE TIME INTERVAL BETWEEN SUCCESSIVE MEASUREMENTS',
       1'(OF ALL CHANNELS)',/,1X,'FOR THIS CELL(TEMPORARY: ADJUSTED',
       2' BY PROGRAM)',/,1X,'MINS,SECS,TICKS')
       RETURN
8       WRITE(NN1,2100)
2100    FORMAT(1X,'INDICATE CHANNEL(1-5) TO BE WATCHED FOR REGULATION ',
       1'OF DATA-TAKING RATE',/,1X,'ON DISCHARGE AND ON CHARGE, AND ',
       2'GIVE(IN APPROPRIATE UNITS: V,A OR DEG C) ',/,1X,'THE MINIMAL',
       3' STEP WHICH SHOULD CAUSE A CHANGE OF RATE.',/,1X,'(NOTE: ',
       4'FOR THE FIRST PART OF A CONSTANT VOLTAGE-LIMITED CURRENT ',
       5'CHARGE,',/,1X,'SELECT A VOLTAGE CHANNEL TO BE WATCHED. AS THE',
       6' SECOND PART STARTS, THE CHANNEL',/,1X,'TO BE FOLLOWED IS ',
       7'AUTOMATICALLY SET AS CHANNEL 4)',/,1X,
```

```
     8'INTEGER(1-5),REAL   FOR DISCHARGE',/,1X,
     9'INTEGER(1-5),REAL   FOR CHARGE',/,1X,'(USUAL CASE IS',/,1X,
     1'1,.01',/,1X,'1,.01)')
     RETURN
9    WRITE(NN1,2200)
2200 FORMAT(1X,'DOES CELL INITIALLY HAVE TO BE CHARGED OR DISCHARGED ?',
     1/,1X,'TYPE 1(CH) OR 0(D)')
     RETURN
10   WRITE(NN1,2300)
     WRITE(NN1,2400)
2300 FORMAT(1X,'DISCHARGE')
2400 FORMAT(1X,'CONTROL MODES :',/,3X,'1: CONSTANT CURRENT',/,3X,
     1'2: CONSTANT CURRENT, ONE DISCHARGE',/,3X,'3: CONSTANT ',
     2'CURRENT, TWO CELLS IN SERIE',/,3X,'4: CONSTANT POWER',
     3/,3X,'5: CONSTANT POWER, ONE DISCHARGE',/,3X,'6: CONSTANT ',
     4'POWER, TWO CELLS IN SERIE',/,3X,'7: PULSED CURRENT',
     5/,3X,'8: PULSED CURRENT, ONE DISCHARGE',/,3X,'9: PULSED ',
     6'CURRENT, TWO CELLS IN SERIE')
     RETURN
11   WRITE(NN1,2500)
     WRITE(NN1,2600)
```

```
2500    FORMAT(1X,'CHARGE')
2600    FORMAT(1X,'CONTROL MODES :',/,3X,'1: CONSTANT CURRENT',/,3X,
        1'2: CONSTANT CURRENT, ONE CHARGE',/,3X,'3: CONSTANT CURRENT, ',
        2'TWO CELLS IN SERIE',/,3X,'4: CONSTANT VOLTAGE-LIMITED CURRENT',
        3/,3X,'5: CV-LC, ONE CHARGE',/,3X,'6: CV-LC, TWO CELLS IN ',
        4'SERIE',/,3X,'7: PULSED CURRENT',/,3X,'8: PULSED CURRENT, ',
        5'ONE CHARGE',/,3X,'9: PULSED CURRENT, TWO CELLS IN SERIE')
        RETURN
12      WRITE(NN1,2700)
2700    FORMAT(1X,'CONTROL MODE ? TYPE 0 IF NONE')
        RETURN
13      WRITE(NN1,2800)
2800    FORMAT(1X,'IF YOU WANT TO SPECIFY A TIME LIMIT FOR THIS MODE,',
        1/,1X,'NOT A CHARGE LIMIT, TYPE Y, OTHERWISE N')
        RETURN
14      WRITE(NN1,2900)
2900    FORMAT(1X,'GIVE DURATION OF MODE: HRS,MINS,SECS,TICKS',/,1X,
        1'MAXIMUM IS 24 HRS')
        RETURN
15      WRITE(NN1,2950)
2950    FORMAT(1X,'GIVE TIME AT OPEN-CIRCUIT DURING WHICH THE PROGRAM',
        1/,1X,'MUST CONTINUE TO TAKE MEASUREMENTS ON THE CELL',/,1X,
        2'HRS,MINS,SECS,TICKS  (0,0,0,0 IS A VALID ANSWER)')
        RETURN
16      WRITE(NN1,3000)
3000    FORMAT(1X,'MAXIMUM NUMBER OF A-H TO BE PASSED ON DISCHARGE:',
        1' (REAL)')
```

```
      RETURN
17    WRITE(NN1,3100)
3100  FORMAT(1X,'NUMBER OF A-H TO BE PASSED ON CHARGE: (REAL)',/,1X,
     1'NOTE: FOR A MULTIPLE CYCLE OPERATION, THIS VALUE WILL BE ',
     2/,1X,'CHANGED BY THE PROGRAM AS CYCLING GOES ON')
      RETURN
18    WRITE(NN1,3200)
3200  FORMAT(1X,'EXCESS FACTOR, WHICH MULTIPLIES A-HRS PASSED ON',
     1' DISCHARGE',/,1X,'TO DEFINE THE A-HRS FOR THE NEXT CHARGE:',
     2' (REAL)')
      RETURN
19    WRITE(NN1,3300)
3300  FORMAT(1X,'DURATION OF OPEN-CIRCUIT : HRS,MINS,SECS,TICKS')
      RETURN
20    IF(IND.EQ.1) WRITE(NN1,2300)
      IF(IND.EQ.2) WRITE(NN1,2500)
      WRITE(NN1,3400)
```

```
3400    FORMAT('+',T11,': LIMITS. GIVE CHANNEL(1-5) & LIMIT(REAL)',
        1/,1X,'+ SIGN ON CHANNEL NUMBER IS UPPER LIMIT',
        2/,1X,'- SIGN ON CHANNEL NUMBER IS LOWER LIMIT',/,1X,'IF ALL ',
        3'NECESSARY LIMITS ENTERED FOR THIS MODE, TYPE 0,0.',
        4/,1X,'FOR EXAMPLE : 2 LIMITS SPECIFIED, UPPER LIMIT 2. V ON ',
        5'CELL VOLTAGE',/,1X,'AND LOWER LIMIT .05 A ON CURRENT :',
        6/,1X,'ENTER : 1,2.',/,8X,'-4,.05',/,9X,'0,0.',/,1X,
        7'NOTE: NEGATIVE LIMIT VALUES ARE ALLOWED !')
        RETURN
21      WRITE(NN1,3500) IND
3500    FORMAT(1X,'LIMIT ',I2,' :CHANNEL,VALUE')
        RETURN
22      IF(IND.EQ.1) WRITE(NN1,2300)
        IF(IND.EQ.2) WRITE(NN1,2500)
        WRITE(NN1,3600)
3600    FORMAT('+',T11,': CURRENT CONTROL VALUE (REAL)')
        RETURN
23      WRITE(NN1,3700)
3700    FORMAT(1X,'GIVE VOLTAGE CHANNEL(1-3) TO BE USED FOR POWER',
        1' REGULATION',/,1X,'AND GIVE VALUE(REAL,IN WATTS) : CHANNEL',
        2',VALUE')
        RETURN
24      WRITE(NN1,3800)
3800    FORMAT(1X,'CONSTANT VOLTAGE-LIMITED CURRENT MODE',/,1X,
        1'GIVE, FIRST, VALUE OF CURRENT FOR START OF CHARGE: (REAL)')
        RETURN
25      WRITE(NN1,3900)
```

```
3900    FORMAT(1X,'THEN, GIVE VOLTAGE CHANNEL(1-3) TO BE CONTROLLED, '
        1' AND VALUE(REAL)',/,1X,'TYPE : CHANNEL,VALUE')
        RETURN
26      WRITE(NN1,4000) IND
4000    FORMAT(1X,'THIS CELL(',I2,') IS CONNECTED TO ANOTHER ONE,',
        1/,1X,'TYPE IT"S NUMBER')
        RETURN
27      WRITE(NN1,4100)
4100    FORMAT(1X,'TIME INTERVAL BETWEEN SUCCESSIVE CURRENT CONTROLS:',/,
        11X,'SECS,TICKS')
        RETURN
        END
1
```

```
      SUBROUTINE MATIN
      COMMON/PASS/A,B,JT,VAL,QO,Q
      INTEGER*4 JT(10)
      LOGICAL*1 ANS
      DIMENSION VAL(14),QO(2),Q(2)
      INTEGER A(22),B(14)
      NN1=7
      IDUM=0
      JMAX=16
      WRITE(NN1,900)
900   FORMAT(1X,'DEFAULT # OF CELLS IS 16. CHANGE ? Y OR N')
      ACCEPT 1000,ANS
1000  FORMAT(A1)
      IF(ANS.NE.'Y') GOTO 3
      WRITE(NN1,1100)
1100  FORMAT(1X,'NUMBER OF CELLS ?')
      ACCEPT *,JMAX
      WRITE(NN1,1150)
1150  FORMAT(1X,'YOU CANNOT CHANGE THE DEFAULT',/,'UNLESS SOME ',
     1'STATEMENTS ARE REMOVED FROM SUBROUTINE MATIN',/,1X,
     2'THE NUMBER OF CELLS WILL REMAIN EQUAL TO 16')
      JMAX=16
3     OPEN(UNIT=10,NAME='VOL:FMAT.UNF',TYPE='UNKNOWN',ACCESS='DIRECT',
     1RECORDSIZE=1,ERR=5,ASSOCIATEVARIABLE=NPOINT)
```

```
        GOTO 10
5       WRITE(NN1,1200)
1200    FORMAT(1X,'ERROR IN FILE OPENING')
        STOP
10      WRITE(10'1)JMAX,IDUM
        WRITE(NN1,1300)
1300    FORMAT(1X,'CELL NUMBER ?')
        ACCEPT *,JCEL
        DO 115 J=1,JMAX
C       if this is the cell, jump
        IF(JCEL.EQ.J) GOTO 15
C       increment the pointer in FMAT.UNF
        NPOINT=NPOINT+49
        WRITE(NN1,1999) NPOINT
1999    FORMAT(1X,'NPOINT=',I3)
        GOTO 115
C       flag for cell in operation
15      A(1)=1
        DO 16 K=2,22
```

```
C       initialize the A array
16      A(K)=0
C       flag for storage on the data-file
        A(4)=3
        DO 17 K=1,14
C       initialize B array
17      B(K)=0
C       initialize timers
        II=JICVT(0,JT(1))
        II=JICVT(0,JT(2))
        II=JICVT(0,JT(3))
        II=JICVT(0,JT(9))
        CALL JTIME(24,0,0,0,JT(5))
        CALL JJCVT(JT(5))
        CALL JTIME(24,0,0,0,JT(6))
        CALL JJCVT(JT(6))
        CALL JTIME(24,0,0,0,JT(7))
        CALL JJCVT(JT(7))
        CALL JTIME(0,1,0,0,JT(10))
        CALL JJCVT(JT(10))
C       initialize the Q and Q0 arrays
        Q0(1)=100.
        Q0(2)=100.
        Q(1)=0.
        Q(2)=0.
C       ask for C.C #, calibration, full-scale current
        CALL ASK(1,J)
```

```
      ACCEPT *,A(9)
      CALL ASK(2,0)
      ACCEPT *,VAL(3)
      CALL ASK(3,0)
      ACCEPT *,A(10)
C     ask for # active channels, A/D channels and ranges
      CALL ASK(4,0)
      ACCEPT *,A(12)
      CALL ASK(5,0)
      DO 20 I=1,5
      L=I-1
      CALL ASK(6,I)
20    ACCEPT *,A(13+2*L),A(14+2*L)
C     ask for DT-measurement, convert to INT*4
      CALL ASK(7,0)
      ACCEPT *,IMIN,ISEC,ITICK
      CALL JTIME(0,IMIN,ISEC,ITICK,JT(4))
      CALL JJCVT(JT(4))
C     ask for channels which regulate data-taking rate, and steps
C     which change the rate
      CALL ASK(8,0)
      DO 25 I=1,2
      ACCEPT *,B(I),VAL(I)
```

```
25      IF(B(I).EQ.4) VAL(I)=VAL(I)*5./A(10)
26      CALL ASK(9,0)
        ACCEPT *,A(3)
C       main  do loop, 1 is ds, 2 is ch
        DO 110 I=1,2
        GOTO(30,35)I
C       ask for ds control mode
30      CALL ASK(10,0)
        GOTO 40
C       ask for ch control mode
35      CALL ASK(11,0)
40      CALL ASK(12,0)
        ACCEPT *,B(I+2)
C       if mode defined (not = 0), jump to 45
        IF(B(I+2).NE.0) GOTO 45
C       if mode = 0, but specified as initial mode -- error
        IF(A(3).EQ.(I-1)) GOTO 50
C       if ds specification and not defined (= 0), jump to end of DO loop
        IF(I.EQ.1) GOTO 110
C       if ch specification and not defined (= 0), and ds mode was spe-
C       cified as single, jump to end of DO loop
        IF(B(3).EQ.2.OR.B(3).EQ.5.OR.B(3).EQ.8) GOTO 110
C       we have a non defined ch and a non single mode ds -- error
        GOTO 50
C       the following tests are for a mode not = 0
45      IZERO=0
        ISING=0
```

```
C       if single mode specified, set ISING
        IF(B(I+2).EQ.2.OR.B(I+2).EQ.5.OR.B(I+2).EQ.8) ISING=1
C       if ISING set, but the mode is not the initial mode -- error
        IF(ISING.EQ.1.AND.A(3).NE.(I-1)) GOTO 50
C       if ds specification, no more tests needed at this point
        IF(I.EQ.1.AND.ISING.EQ.0) GOTO 60
        IF(I.EQ.1.AND.ISING.EQ.1) GOTO 55
C       the following tests get executed only after the ch specification
C       if ds mode is single -- error
        IF(B(3).EQ.2.OR.B(3).EQ.5) GOTO 50
C       if ds mode = 0, set IZERO
        IF(B(3).EQ.0) IZERO=1
C       if IZERO & ISING not both set or 0 -- error
        IF(IZERO.NE.ISING) GOTO 50
C       if ISING set, jump to 55
        IF(ISING.EQ.1) GOTO 55
        MSERIE=0
        NSERIE=0
C       if 2 cells in serie for ds, set MSERIE
        IF(B(3).EQ.3.OR.B(3).EQ.6.OR.B(3).EQ.9) MSERIE=1
C       if 2 cells in serie for ch, set NSERIE
        IF(B(4).EQ.3.OR.B(4).EQ.6.OR.B(4).EQ.9) NSERIE=1
C       if MSERIE & NSERIE both set or 0, then OK
        IF(MSERIE.EQ.NSERIE) GOTO 60
```

```
1
C       error
50      WRITE(NN1,2800)
2800    FORMAT(1X,'WRONG COMBINATION OF CONTROL MODES')
        GOTO 26
C       the following questions are for single ch or ds
C       time limit or A-hr limit ?
55      CALL ASK(13,0)
        ACCEPT 1000,ANS
        IF(ANS.NE.'Y') GOTO 56
C       ask for time limit, convert to INT*4 time
        CALL ASK(14,0)
        ACCEPT *,IHR,IMIN,ISEC,ITICK
        CALL JTIME(IHR,IMIN,ISEC,ITICK,JT(7-I))
        CALL JJCVT(JT(7-I))
        GOTO 58
C       if ch, skip
56      IF(I.EQ.2) GOTO 57
C       ask for ds A-hrs, convert to V-hrs, using full-scale current
        CALL  ASK(16,0)
        ACCEPT *,Q0(1)
        Q0(1)=Q0(1)*5./A(10)
        GOTO 58
C       ask for ch A-hrs, convert to V-hrs, using full-scale current
57      CALL ASK(17,0)
        ACCEPT *,Q0(2)
        Q0(2)=Q0(2)*5./A(10)
```

```
C       ask for o-c time after single mode, convert to INT*4 time
58      CALL ASK(15,0)
        ACCEPT *,IHR,IMIN,ISEC,ITICK
        CALL JTIME(IHR,IMIN,ISEC,ITICK,JT(7))
        CALL JJCVT(JT(7))
C       single mode, skip next questions
        GOTO 80
C       the following questions are for non-single ch or ds
C       if ch, skip
60      IF(I.EQ.2) GOTO 65
C       ask for ds A-hrs, convert to v-hrs, using full-scale current
        CALL ASK(16,0)
        ACCEPT*,QO(1)
        QO(1)=QO(1)*5./A(10)
C       ds, skip next
        GOTO 80
C       if starting mode is ds, skip
65      IF(A(3).EQ.0) GOTO 70
C       ask for ch A-hrs, convert to V-hrs, using full-scale current
        CALL ASK(17,0)
        ACCEPT *,QO(2)
        QO(2)=QO(2)*5./A(10)
```

```
C       ask for excess factor
70      CALL ASK(18,0)
        ACCEPT *,VAL(4)
C       if starting mode is ch, initialize Q(1): this is needed because,
C       in the real-time program, at the start of every ch, the A-hrs
C       of the previous ds are multiplied by the excess factor, to de-
C       fine the maximum number of A-hrs to pass on ch
        IF(A(3).EQ.1) Q(1)=Q0(2)/VAL(4)
C       if starting mode is ds, initialize Q0(2) to Q0(1) (safety)
        IF(A(3).EQ.0) Q0(2)=Q0(1)
C       ask for o-c time, convert to INT*4
75      CALL ASK(19,0)
        ACCEPT *,IHR,IMIN,ISEC,ITICK
        CALL JTIME(IHR,IMIN,ISEC,ITICK,JT(7))
        CALL JJCVT(JT(7))
C       ask for limits
80      CALL ASK(20,I)
C       ds limits are numbered 1 to 3, ch limits 4 to 6
        DO 85 KK=1+(I-1)*3,3+(I-1)*3
        CALL ASK(21,KK)
        ACCEPT *,B(4+KK),VAL(4+KK)
C       if not limit on current channel, skip
        IF(IABS(B(4+KK)).NE.4) GOTO 82
C       on ch, the current is < 0, so switch the sign of the limit value
        VAL(4+KK)=(3-2*I)*VAL(4+KK)
C       on ch, an upper limit on the current becomes a lower limit, and
C       vice-versa
```

```
C       (< x amp becomes > -x amp)
        IF(VAL(4+KK).LT.0.) B(4+KK)=-B(4+KK)
C       if limit channel is entered 0, then no more limits specified
82      IF(B(4+KK).EQ.0) GOTO 90
85      CONTINUE
C       if not constant current ch or ds, skip
90      IF(B(I+2).GT.3) GOTO 95
C       ask for current, convert to volts, using full-scale current
        CALL ASK(22,I)
        ACCEPT *,VAL(10+I)
        VAL(10+I)=VAL(10+I)*5./A(10)
        GOTO 110
C       if external modes, skip
95      IF(B(2+I).GT.6) GOTO 110
        GOTO(100,105) I
C       constant power ds, ask for power level
100     CALL ASK(23,0)
        ACCEPT *,B(13),VAL(13)
        GOTO 110
```

```
C       CVLC ch, ask for current, convert to V-hrs, ask for voltage
105     CALL ASK(24,0)
        ACCEPT *,VAL(12)
        VAL(12)=VAL(12)*5./A(10)
        CALL ASK(25,0)
        ACCEPT *,B(14),VAL(14)
110     CONTINUE
C       if not external modes, skip
        IF(B(3).LE.6.AND.B(4).LE.6) GOTO 112
C       define channel 5 as cell voltage @ high side of pulse, with a
C       range = range + 100
        A(12)=A(12)+1
        A(21)=A(13)
        A(22)=A(14)+100
        WRITE(NN1,2900) A(21),A(12)
2900    FORMAT(1X,'BECAUSE ONE OF YOUR MODES IS AN EXTERNAL PULSE MOD',
       1'E, CHANNEL 5',/,1X,'HAS BEEN DECLARED ACTIVE AND ASSIGNED ',
       2'THE SAME A/D CHANNEL (',I2,')',/,1X,'AS CHANNEL 1. IT WILL ',
       3'MEASURE THE CELL VOLTAGE AT THE HIGH SIDE',/,1X,'OF THE PUL',
       4'SE. YOU NOW HAVE ',I2,' CHANNELS ACTIVE.')
C       if cell in serie, ask which one
112     IF(MSERIE.EQ.1) CALL ASK(26,J)
        IF(MSERIE.EQ.1) ACCEPT *,A(11)
C       ask for DT-control, convert to INT*4 time
        CALL ASK(27,0)
        ACCEPT *,ISEC,ITICK
        CALL JTIME(0,0,ISEC,ITICK,JT(8))
```

```
      CALL JJCVT(JT(8))
C     skip the first 3 records of FMAT.UNF
      NPOINT=NPOINT+3
C     write all the arrays out to FMAT.UNF
      DO 200 KK=1,21,2
200   WRITE(10'NPOINT) A(KK),A(KK+1)
      DO 210 KK=1,13,2
210   WRITE(10'NPOINT) B(KK),B(KK+1)
      DO 220 KK=1,10
220   WRITE(10'NPOINT) JT(KK)
      DO 230 KK=1,14
230   WRITE(10'NPOINT) VAL(KK)
      DO 240 KK=1,2
240   WRITE(10'NPOINT) Q0(KK)
      DO 250 KK=1,2
250   WRITE(10'NPOINT) Q(KK)
      TYPE 1999,NPOINT
115   CONTINUE
C     write record number 786 of the file, to define its length
      WRITE(10'(NPOINT+49)) IDUM
      CLOSE(UNIT=10)
      RETURN
      END
```

```
1

      SUBROUTINE MODIF(ICHANG)
      COMMON/PASS/A,B,JT,VAL,QO,Q
      LOGICAL*1 ANS,MODENM(9,2),MODELM(5,2)
      INTEGER A(22),B(14),MSG(58),RESP(81),AREA(4),LIST(20)
      INTEGER*4 JT(8),JFAC
      DIMENSION VAL(14),QO(2),Q(2),QQO(2)
      DATA MODENM/'D','I','S','C','H','A','R','G','E','C','H','A','R',
     1'G','E',' ',' ',' '/
      DATA MODELM/'U','P','P','E','R','L','O','W','E','R'/
      EQUIVALENCE (RESP(39),JT(4))
      EQUIVALENCE (RESP(49),VAL(1))
      EQUIVALENCE (RESP(77),QO(1))
      EQUIVALENCE(MSG(31),JFAC)
      NN1=7
      NN2=6
      LCHANG=0
      II=JICVT(0,JFAC)
      WRITE(NN1,1000)
1000  FORMAT(1X,'CELL NUMBER ?')
      READ(5,*) JCEL
      IF(JCEL.LT.1.OR.JCEL.GT.16) TYPE *,'INVALID CELL NUMBER'
```

```
      IF(JCEL.LT.1.OR.JCEL.GT.16) RETURN
C     if EX mode, no message
      IF(ICHANG.EQ.-1) GOTO 8
      IF(ICHANG.EQ.0) MSG(1)='DS'
      IF(ICHANG.EQ.1) MSG(1)='CH'
      MSG(2)=JCEL
C     send message
      NFG=ISDATW(MSG,2)
C     if no foreground, jump
      IF(NFG.EQ.1) GOTO 5
C     wait for response
      II=IRCVDW(RESP,81)
      IF(RESP(2).EQ.'OK') GOTO 1
      WRITE(NN1,1100)
1100  FORMAT(1X,'COMMUNICATION PROBLEM')
      WRITE(NN1,1200)
1200  FORMAT(1X,'CHANGE PROCEDURE ABORTED')
      RETURN
```

```
C       decode response from REAL-TIME program (see equivalence
C       statements)
1       DO 2 K=1,22
2       A(K)=RESP(2+K)
        DO 3 K=1,14
3       B(K)=RESP(24+K)
        ICYCLE=RESP(81)
        GOTO 35
C       no foreground, read from GMAT.UNF
5       WRITE(NN1,1300)
1300    FORMAT(1X,'NO FOREGROUND JOB ACTIVE. DIRECT COMMUNICATION WITH',
        1' GMAT.UNF',/)
        IUN=12
        OPEN(UNIT=12,NAME='VOL:GMAT.UNF',TYPE='OLD',ACCESS='DIRECT',
        1RECORDSIZE=1,ERR=1395,ASSOCIATEVARIABLE=NPOINT)
        GOTO 9
1395    WRITE(NN1,1400)
1400    FORMAT(1X,'ERROR IN FILE OPENING')
        STOP
C       EX mode: read from FMAT.UNF
8       IUN=10
        OPEN(UNIT=10,NAME='VOL:FMAT.UNF',TYPE='OLD',ACCESS='DIRECT',
        1RECORDSIZE=1,ERR=1395,ASSOCIATEVARIABLE=NPOINT)
9       NPOINT=5+49*(JCEL-1)
C       read file (FMAT.UNF or GMAT.UNF)
        DO 10 I=1,21,2
10      READ(IUN'NPOINT) A(I),A(I+1)
```

```
        DO 15 I=1,13,2
15      READ(IUN'NPOINT) B(I),B(I+1)
        NPOINT=NPOINT+3
        DO 20 I=4,8
20      READ(IUN'NPOINT) JT(I)
        NPOINT=NPOINT+2
        DO 25 I=1,14
25      READ(IUN'NPOINT) VAL(I)
        DO 30 I=1,2
30      READ(IUN'NPOINT) QO(I)
C       if EX mode, no cycle information
        IF(ICHANG.EQ.-1) GOTO 35
        NPOINT=785+JCEL
        READ(12'NPOINT) ICYCLE,MCYCLE
35      WRITE(NN1,1500)
1500    FORMAT(1X,'DISPLAY OF PARAMETERS FOLLOWS. IN THE CHANGE MODE, ',
        1'ONLY THE ONES',/,1X,'REFERENCED WITH A NUMBER CAN BE CHANGED',
        2//,1X,'DEFAULT PRINTOUT IS ON TT: . DO YOU WANT LS: ? Y OR N')
        READ(5,1600) ANS
1600    FORMAT(A1)
        IF(ANS.EQ.'Y') NN=NN2
        IF(ANS.NE.'Y') NN=NN1
```

```
C       new page
        IF(NN.EQ.NN2) WRITE(NN,1650)
1650    FORMAT('1')
C
C             cycling conditions
C
C       if EX mode, no cycle information
        IF(ICHANG.EQ.-1) GOTO 37
        IF(A(1).EQ.0) WRITE(NN,1700) JCEL,ICYCLE
        IF(A(1).EQ.1) WRITE(NN,1800) JCEL,ICYCLE
1700    FORMAT(1X,'CELL ',I2,' STOPPED RUNNING AT CYCLE ',I3)
1800    FORMAT(1X,'CELL ',I2,' IS RUNNING, CYCLE ',I3)
37      IF(A(2).EQ.0) WRITE(NN,1900) JCEL,(MODENM(KK,A(3)+1),KK=1,9)
        IF(A(2).EQ.1) WRITE(NN,2000) (MODENM(KK,A(3)+1),KK=1,9)
1900    FORMAT(1X,'CELL ',I2,' IS IN OPEN-CIRCUIT. NEXT MODE IS ',9A1,/)
2000    FORMAT(1X,'PRESENT MODE IS ',9A1,/)
C
C             scratch-data-file information
C
C       if EX mode, none
        IF(ICHANG.EQ.-1) GOTO 38
        WRITE(NN,2050) A(7)
2050    FORMAT(1X,'LAST BLOCK ON SCRATCH-FILE THAT HAS BEEN WRITTEN TO',
        1' IS BLOCK NUMBER ',I3,/)
C
C             current controller information
C
```

```
38      WRITE(NN,2100) A(9),A(10),VAL(3)
2100    FORMAT(1X,'CURRENT CONTROLLER ',I1,'. FULL-SCALE CURRENT IS ',
       1I2,' A,'/,5X,'WITH CALIBRATION FACTOR ',F10.5,T65,'#  1',/)
        IF(A(11).NE.0) WRITE(NN,2200) A(11)
2200    FORMAT(1X,'CELL IN SERIE WITH CELL ',I2,/)
        WRITE(NN,2300) A(12),(KK,A(11+2*KK),A(12+2*KK),KK=1,5)
2300    FORMAT(1X,I2,' A/D CHANNELS ACTIVE',T65,'#  2',
       15(/,5X,'CHANNEL ',I2,': A/D CHANNEL ',I3,/,17X,'RANGE ',
       2'NUMBER ',I3),/)
C
C               data taking rate adjustment information
C
        DO 40 K=1,2
        VVAL=VAL(K)
C       if current channel followed for rate adjustment, convert to
C       Amperes
        IF(B(K).EQ.4) VVAL=VVAL*A(10)/5.
40      WRITE(NN,2400) (MODENM(KK,K),KK=1,9),K,B(K),VVAL
2400    FORMAT(1X,'ON ',9A1':',T65,'# 2',I1,
       1/,5X,'CHANNEL WATCHED FOR REGULATION OF',
       1' DATA-TAKING RATE IS ',I2,/,5X,'INTERVAL THAT REGULATES THE',
       2' RATE IS ',F10.5,/,10X,' (IN APPROPRIATE UNITS)',/)
        WRITE(NN,2500) (MODENM(KK,1),KK=1,9),B(3)
        WRITE(NN,2500) (MODENM(KK,2),KK=1,9),B(4)
```

```
2500    FORMAT(1X,9A1,' MODE IS ',I1,/)
1
C               limit information
C
        IJUMP=0
45      DO 55 L=1,2
C       if call from somewhere below and this is not the right mode,
C       jump
        IF(IJUMP.NE.0.AND.L.NE.IJUMP) GOTO 55
        KKK=2+L
        DO 50 I=1,3
C       get limit channel & value
        IBB=B(1+3*L+I)
        VVAL=VAL(1+3*L+I)
C       if no more limits for the mode, jump
        IF(IABS(IBB).EQ.0) GOTO 55
        IF(I.EQ.1) WRITE(NN,2600) KKK
2600    FORMAT(1X,T65,'# 2',I1)
C       limit number (1,2,3 for ds, 4,5,6 for ch)
        II=I+3*(L-1)
C       upper (+) or lower (-) limit ?
        ISIGN=IABS(IBB)/IBB
C       if lower, set ISIGN=2. If upper, ISIGN=1.
        IF(ISIGN.EQ.-1) ISIGN=2
C       current is a special case : on charge, the current is always
C       negative, therefore, a lower limit becomes an upper limit
C       and vice-versa
```

```
C       e.g. charge current smaller than .2 A becomes : larger than
C       - .2 A
        IF(L.EQ.1.OR.IABS(IBB).NE.4) GOTO 50
C       for a current limit, show only the absolute value
        ISIGN=3-ISIGN
        VVAL=ABS(VVAL)
50      WRITE(NN,2700) (MODELM(KK,ISIGN),KK=1,5),II,(MODENM(KK,L),KK=1,9),
        1IABS(IBB),VVAL
2700    FORMAT(1X,5A1,' LIMIT ',I1,' ON ',9A1,': CHANNEL ',I2,
        1/,29X,'VALUE ',F10.5)
55      WRITE(NN,2800)
2800    FORMAT(1X)
C       if call from somewhere below, jump back
        IF(IJUMP.NE.0) GOTO 92
C
C               timer information
C
        JJUMP=0
57      DO 60 L=4,8
C       if call from somewhere below, and this is not the right timer,
C       jump
        IF(JJUMP.NE.0.AND.L.NE.JJUMP) GOTO 60
```

```
C       don't show charge or discharge time, unless it's a single
C       half-cycle mode
        IF((L.EQ.5.AND.(B(4).NE.2.AND.B(4).NE.5.AND B(4).NE.8)).OR.
        (L.EQ.6.AND.(B(3).NE.2.AND.B(3).NE.5.AND.B(3).NE.8))) GOTO 60
        CALL JJCVT(JT(L))
        CALL CVTTIM(JT(L),IHR,IMIN,ISEC,ITICK)
        CALL JJCVT(JT(L))
C       if timer = 24 hrs exactly, then it has no significance
        IF(IHR.EQ.24.AND.IMIN.EQ.0.AND.ISEC.EQ.0.AND.ITICK.EQ.0) GOTO 60
        IF(L.EQ.4) WRITE(NN,2900)
        IF(L.EQ.5) WRITE(NN,3000)
        IF(L.EQ.6) WRITE(NN,3100)
        IF(L.EQ.7) WRITE(NN,3200)
        IF(L.EQ.8) WRITE(NN,3300)
        WRITE(NN,3400) IHR,IMIN,ISEC,ITICK
2900    FORMAT(1X,'DT-MEASUREMENT',T65,'# 31')
3000    FORMAT(1X,'DT-CHARGE')
3100    FORMAT(1X,'DT-DISCHARGE')
3200    FORMAT(1X,'DT-OPEN-CICUIT',T65,'# 32')
3300    FORMAT(1X,'DT-CONTROL',T65,'# 33')
3400    FORMAT('+',T20,I2,':',I2,':',I2,':',I2,/)
60      CONTINUE
C       if call from somewhere below, jump back
        IF(JJUMP.NE.0) GOTO 110
C
C               excess factor information
C
```

```
C       if single half-cycle, no excess factor
        IF(B(3).NE.2.AND.B(3).NE.5.AND.B(3).NE.8.AND.B(4).NE.2
        1.AND.B(4).NE.5.AND.B(4).NE.8) WRITE(NN,3500) VAL(4)
3500    FORMAT(1X,'CHARGE EXCESS FACTOR ',F10.5,T65,'# 41',/)
C
C               current information
C
        DO 65 K=1,2
C       if the control mode is not constant current, jump
        IF((K.EQ.1.AND.(B(3).GE.4.OR.B(3).EQ.0)).OR.(K.EQ.2.AND.
        1(B(4).GE.7.OR.B(4).EQ.0))) GOTO 65
C       convert to Ampere
        VAL(10+K)=VAL(10+K)*A(10)/5.
        WRITE(NN,3600) (MODENM(KK,K),KK=1,9),VAL(10+K),K+1
3600    FORMAT(1X,'CURRENT ON ',9A1,3X,F10.5,' A',T65,'# 4',I1,/)
C       convert back to Volt
        VAL(10+K)=VAL(10+K)*5./A(10)
65      CONTINUE
C
C               power information
C
C       if not constant power mode, skip
        IF(B(3).EQ.4.OR.B(3).EQ.5.OR.B(3).EQ.6) WRITE(NN,3700)
        1VAL(13),B(13)
```

```
3700    FORMAT(1X,'POWER ON DISCHARGE ',2X,F10.5,' W',/,5X,'WITH ',
       1'VOLTAGE CHANNEL ',I1,/)
C
C               voltage information
C
C       if not constant voltage-limited current mode, skip
        IF(B(4).GT.3.AND.B(4).LT.7) WRITE(NN,3800) VAL(14),B(14)
3800    FORMAT(1X,'VOLTAGE PLATEAU ON CHARGE ',2X,F10.5,' V',/,5X,
       1'CHANNEL ',I1,/)
C
C               A-hr information
C
        DO 70 K=1,2
C       if control mode is 0, jump
        IF((K.EQ.1.AND.B(3).EQ.0).OR.(K.EQ.2.AND.B(4).EQ.0)) GOTO 70
C       convert V-hr to A-hr
        QQ0(K)=Q0(K)*A(10)/5.
        WRITE(NN,3900) (MODENM(KK,K),KK=1,9),QQ0(K)
3900    FORMAT(1X,'MAXIMUM A-HRS TO BE PASSED ON ',9A1,2X,F10.5)
C       skip if charge A-hrs
        IF(K.EQ.1) WRITE(NN,4000)
4000    FORMAT('+',T65,'# 51')
        WRITE(NN,2800)
70      CONTINUE
        REWIND NN
C       unless in the 'CH' mode, done
        IF(ICHANG.NE.1) GOTO 185
```

```
      WRITE(NN1,4100)
4100  FORMAT(1X,'YOU CAN ONLY CHANGE THE PARAMETERS REFERENCED WITH ',
     1'A NUMBER')
      WRITE(NN1,4200)
4200  FORMAT(1X,'DO YOU FEEL LIKE CHANGING SOMETHING ? TYPE Y OR N')
      READ(5,1600) ANS
C     if (more) changes, jump
      IF(ANS.EQ.'Y') GOTO 80
C     if no more changes, and something changed already, jump to
C     where the changes are made permanent
      IF(LCHANG.EQ.1) GOTO 120
C     if no more changes, and nothing changed, abort 'CH' procedure
72    MSG(1)='BD'
C     if no foreground active, jump
      IF(NFG.EQ.1) GOTO 75
C     if foreground active, send abort message
      II=ISDATW(MSG,1)
      II=IRCVDW(RESP,2)
      IF(RESP(2).NE.'BD') WRITE(NN1,1100)
      IF(RESP(2).EQ.'BD') WRITE(NN1,1200)
      RETURN
75    CLOSE(UNIT=12)
      WRITE(NN1,1200)
      RETURN
```

```
C       set flag for 'some changes'
80      LCHANG=1
84      WRITE(NN1,4300) JCEL
4300    FORMAT(1X,'CHANGES FOR CELL ',I2,/,1X,'CONSTRUCT A STRING OF ',
        1'AT MOST 20 NUMBERS (INCLUDING TERMINATOR),',/,1X,'CONTAINING ',
        2'THE REFERENCE NUMBERS OF THE PARAMETERS YOU WANT',/,1X,'TO ',
        3'CHANGE',/,1X,'TERMINATE THE STRING WITH 0/',/,1X,'E.G. ',
        4'2,21,32,0/')
        READ(5,*) LIST
1
C       start a loop
        K=0
85      K=K+1
C       if end of the input string, go back for new display of parameters
        IF(K.EQ.20.OR.LIST(K).EQ.0) GOTO 35
        IF(LIST(K).GE.50) GOTO 89
        IF(LIST(K).GE.40) GOTO 88
        IF(LIST(K).GE.30) GOTO 87
        IF(LIST(K).GE.20) GOTO 86
        GOTO (5501,5502) LIST(K)
86      GOTO (5521,5521,5523,5523) LIST(K)-20
87      GOTO (5531,5532,5533) LIST(K)-30
88      GOTO (5541,5542,5542) LIST(K)-40
89      GOTO (5551) LIST(K)-50
        WRITE(NN1,4400) LIST(K)
4400    FORMAT(1X,'BAD ENTRY : # ',I2)
        GOTO 84
```

```
C
C               current controller information
C
5501    WRITE(NN1,2100) A(9),A(10),VAL(3)
        WRITE(NN1,4500)
4500    FORMAT(1X,'CHANGE ? Y OR N')
        READ(5,1600) ANS
        IF(ANS.NE.'Y') GOTO 85
C       ask for controller specifications
        CALL ASK(1,JCEL)
        READ(5,*) A(9)
        CALL ASK(3,0)
        READ(5,*) A(10)
        CALL ASK(2,0)
        READ(5,*) VAL(3)
        GOTO 85
C
C               A/D information
C
5502    WRITE(NN1,2300) A(12),(KK,A(11+2*KK),A(12+2*KK),KK=1,5)
        WRITE(NN1,4500)
        READ(5,1600) ANS
        IF(ANS.NE.'Y') GOTO 85
```

```
C       ask for A/D information
        CALL ASK(4,0)
        READ(5,*) A(12)
        CALL ASK(5,0)
        DO 90 I=1,5
        CALL ASK(6,I)
90      READ(5,*) A(11+2*I),A(12+2*I)
        GOTO 85
C
C               information on the adjustment of the data taking rate
C
5521    L=LIST(K)-20
        WRITE(NN1,2400) (MODENM(KK,L),KK=1,9),L,B(L),VAL(L)
        WRITE(NN1,4500)
        READ(5,1600) ANS
        IF(ANS.NE.'Y') GOTO 85
C       ask for data-taking rate specifications
        WRITE(NN1,4600)
4600    FORMAT(1X,'ENTER CHANNEL(1-5),INTERVAL(REAL)')
        READ(5,*) B(L),VAL(L)
        IF(B(L).EQ.4) VAL(L)=VAL(L)*5./A(10)
        GOTO 85
C
C               limit information
C
C       jump to previous part of the program, for limit display
C       come back at statement label 92
```

```
5523    IJUMP=LIST(K)-22
        GOTO 45
92      WRITE(NN1,4500)
        READ(5,1600) ANS
        IF(ANS.NE.'Y') GOTO 85
C       ask for limit information
        CALL ASK(20,IJUMP)
        DO 93 I=1+(IJUMP-1)*3,3+(IJUMP-1)*3
93      B(4+I)=0
        DO 100 I=1+(IJUMP-1)*3,3+(IJUMP-1)*3
        CALL ASK(21,I)
        READ(5,*) B(4+I),VAL(4+I)
C       special case for the current (see subroutine MATIN)
        IF(IABS(B(4+I)).NE.4) GOTO 95
        VAL(4+I)=(3-2*IJUMP)*VAL(4+I)
        IF(VAL(4+I).LT.0.) B(4+I)=-B(4+I)
95      IF(B(4+I).EQ.0) GOTO 105
100     CONTINUE
```

```
C       reset jump flag
105     IJUMP=0
        GOTO 85
C
C               timer information
C
C       jump to previous section of the program, for timer display
C       come back at statement label 110
5531    JJUMP=4
        GOTO 57
5532    JJUMP=7
        GOTO 57
5533    JJUMP=8
        GOTO 57
110     WRITE(NN1,4500)
        READ(5,1600) ANS
        IF(ANS.NE.'Y') GOTO 85
C       enter timer information
        WRITE(NN1,4700)
4700    FORMAT(1X,'ENTER HR,MIN,SEC,TICK')
        READ(5,*) IHR,IMIN,ISEC,ITICK
        CALL JTIME(IHR,IMIN,ISEC,ITICK,JT(JJUMP))
        CALL JJCVT(JT(JJUMP))
C       if JT(4,J) not modified, or no foreground running, jump
        IF(JJUMP.NE.4.OR.NFG.EQ.1) GOTO 112
C       compute factor that cuts DTstore (JT(4,J))
        CALL FACADJ(JFAC,IHR,IMIN,ISEC,ITICK)
```

```
112     JJUMP=0
        GOTO 85
C
C               excess factor information
C
5541    WRITE(NN1,3500) VAL(4)
        WRITE(NN1,4500)
        READ(5,1600) ANS
        IF(ANS.NE.'Y') GOTO 85
        CALL ASK(18,0)
        READ(5,*) VAL(4)
        GOTO 85
C
C               current information
C
5542    L=LIST(K)-31
C       convert to Ampere
        VAL(L)=VAL(L)*A(10)/5.
        WRITE(NN1,3600) (MODENM(KK,LIST(K)-41),KK=1,9),VAL(L),LIST(K)-40
        WRITE(NN1,4500)
        READ(5,1600) ANS
        IF(ANS.NE.'Y') GOTO 113
        CALL ASK(22,LIST(K)-41)
        READ(5,*) VAL(L)
        IF(A(11).NE.0) WRITE(NN1,4750) JCEL,A(11)
```

```
4750    FORMAT(1X,'CELL ',I2,' IS CONNECTED WITH CELL ',I2,'.',/,1X,
       1'WARNING !!!!! -- CHANGE THE CURRENT FOR THE OTHER CELL, TOO.')
C       convert back to Volt
113     VAL(L)=VAL(L)*5./A(10)
        GOTO 85
C
C               A-hr information
C
5551    WRITE(NN1,3900) (MODENM(KK,1),KK=1,9),QQ0(1)
        WRITE(NN1,4500)
        READ(5,1600) ANS
        IF(ANS.NE.'Y') GOTO 85
C       ask for A-hrs
        CALL ASK(16,0)
        READ(5,*) Q0(1)
C       convert to V-hr
        Q0(1)=Q0(1)*5./A(10)
C       done with changes, go back for new display of parameters
        GOTO 35
120     WRITE(NN1,4800)
4800    FORMAT(1X,'YOU HAVE MADE SOME CHANGES. HOWEVER, THEY HAVEN''T ',
       1'BEEN MADE PERMANENT',/,1X,'YET. DO YOU WANT THAT TO HAPPEN ?',
       2/,1X,'TYPE Y OR N')
        READ(5,1600) ANS
C       changes don't go through, go back to abort point
        IF(ANS.NE.'Y') GOTO 72
        GOTO(125,160) NFG+1
```

```
C       if foreground running, encode message
125     MSG(1)='T1'
        MSG(2)=JCEL
130     DO 135 K=1,14
135     MSG(2+K)=A(8+K)
        MSG(17)=B(1)
        MSG(18)=B(2)
        DO 140 K=1,6
140     MSG(18+K)=B(4+K)
        MSG(25)=RESP(39)
        MSG(26)=RESP(40)
        DO 145 K=1,4
145     MSG(26+K)=RESP(44+K)
        DO 150 K=1,24
150     MSG(32+K)=RESP(48+K)
        MSG(57)=RESP(77)
        MSG(58)=RESP(78)
C       send message
        NFG=ISDATW(MSG,58)
C       if foreground not running (anymore), jump
        IF(NFG.EQ.1) GOTO 160
C       wait for 'OK' response
        II=IRCVDW(RESP,2)
        IF(RESP(2).EQ.'OK') WRITE(NN1,4900)
```

```
4900    FORMAT(1X,'TRANSFER COMPLETED')
        IF(RESP(2).NE.'OK') WRITE(NN1,1100)
        RETURN
C       if no foreground running, write to GMAT.UNF
160     WRITE(NN1,1300)
        NPOINT=9+49*(JCEL-1)
        DO 165 I=9,21,2
165     WRITE(12'NPOINT) A(I),A(I+1)
        WRITE(12'NPOINT) B(1),B(2)
        NPOINT=NPOINT+1
        DO 170 I=5,9,2
170     WRITE(12'NPOINT) B(I),B(I+1)
        NPOINT=NPOINT+5
        DO 175 I=4,8
175     WRITE(12'NPOINT) JT(I)
        NPOINT=NPOINT+2
        DO 180 I=1,12
180     WRITE(12'NPOINT) VAL(I)
        NPOINT=NPOINT+2
        WRITE(12'NPOINT) QO(1)
C       if no foreground running and 'CH' or 'DS' mode, close GMAT.UNF.
C       if 'EX' mode, close FMAT.UNF
185     IF(NFG.EQ.1.OR.ICHANG.EQ.-1) CLOSE(UNIT=IUN)
        IF(ICHANG.EQ.1) WRITE(NN1,4900)
        RETURN
        END
        SUBROUTINE FACADJ(JFAC,IHR,IMIN,ISEC,ITICK)
```

```
      INTEGER*4 JFAC
C     compute the factor that cuts DTstore (JT(4,J))
      FAC=FLOAT(IMIN)+FLOAT(ISEC)/60.+FLOAT(ITICK)/3600.
      IF(FAC.GT.16.) FAC=16.
      FAC=ALOG(FAC*2.)/ALOG(2.)
      IFAC=FAC
      IF((FAC-IFAC).GT.(.5)) IFAC=IFAC+1
      FAC=2.**(FLOAT(IFAC)/2)
      II=JAFIX(FAC,JFAC)
      RETURN
      END
1
```

```
      SUBROUTINE HOLD(IHOLD)
      COMMON/PASS/A,B,JT,VAL,QO,Q
      LOGICAL*1 ANS,MODENM(9,2)
      INTEGER A(22),B(14),MSG(6),RESP(12)
      INTEGER*4 JT(10),JTLEFT
      DIMENSION VAL(14),QO(2),Q(2)
      DATA MODENM/'D','I','S','C','H','A','R','G','E','C','H','A','R',
     1'G','E',' ',' ',' '/
      EQUIVALENCE (RESP(3),JTLEFT)
      EQUIVALENCE (RESP(6),QQO)
      EQUIVALENCE (RESP(8),QQ)
      NN1=20
      NN2=30
      WRITE(NN1,1000)
1000  FORMAT(1X,'CELL NUMBER ?')
      READ(5,*) JCEL
      IF(JCEL.LT.1.OR.JCEL.GT.16) TYPE *,'INVALID CELL NUMBER'
      IF(JCEL.LT.1.OR.JCEL.GT.16) RETURN
      MSG(2)=JCEL
      IF(IHOLD.EQ.1) MSG(1)='HD'
      IF(IHOLD.EQ.0) MSG(1)='ST'
C     send message
      NFG=ISDATW(MSG,2)
```

```
C       if foreground active, jump
        IF(NFG.EQ.0) GOTO 5
C       if no foreground job active, invalid request
1       WRITE(NN1,1100)
1100    FORMAT(1X,'NO FOREGROUND JOB ACTIVE. YOUR REQUEST IS INVALID')
        RETURN
C       wait for response from REAL-TIME program
5       II=IRCVDW(RESP,11)
C       if 'OK', jump
        IF(RESP(2).EQ.'OK') GOTO 20
C       if not 'OK' and not 'NG', jump
        IF(RESP(2).NE.'NG') GOTO 10
C       if 'NG', request aborted
C       convert time left for request that got cancelled
        CALL JJCVT(JTLEFT)
        CALL CVTTIM(JTLEFT,IHR,IMIN,ISEC,ITICK)
        WRITE(NN1,1200) JCEL,IHR,IMIN,ISEC,ITICK
1200    FORMAT(1X,'YOUR REQUEST CAUSED THE CANCELLATION OF A SCHEDU',
       1'LED EVENT FOR CELL ',I2,'.',/,1X,'NO REQUEST AT ALL WILL BE',
       2' EXECUTED FOR THIS CELL.',/,1X,I2,' HRS, ',I2,' MINS, ',I2,
       3' SECS, ',I2,' TICKS REMAINED UNTIL THE EVENT',/,1X,
       4'WOULD HAVE OCCURED.')
        RETURN
```

```
C       if 'NC' response, cell not in operation, request aborted
10      IF(RESP(2).NE.'NC') GOTO 15
        WRITE(NN1,1300)
1300    FORMAT(1X,'THE CELL YOU WANT TO STOP IS NOT IN OPERATION')
        RETURN
C       if can't recognize the response, communication problem
C       request aborted
15      WRITE(NN1,1400) RESP(2)
1400    FORMAT(1X,'COMMUNICATION PROBLEM',/,1X,'CODE RETURNED FROM FG',
        1' IS ',A2)
        RETURN
C       describe the cell status
20      IF(RESP(4).EQ.0) WRITE(NN1,1500) JCEL,RESP(3),
        1(MODENM(KK,2-RESP(5)),KK=1,9)
1500    FORMAT(1X,'CELL ',I2,' IS IN OPEN-CICUIT. THE LAST HALF-CYCLE ',
        1'WAS ',I3,2X,9A1)
        IF(RESP(4).EQ.1) WRITE(NN1,1600) JCEL,RESP(3),
        1(MODENM(KK,RESP(5)+1),KK=1,9),QQ0,QQ
1600    FORMAT(1X,'CELL ',I2,' IS RUNNING ON CYCLE ',I3,' AND IS IN ',
        19A1,' .',/,1X,'OF THE ',F8.5,' A-HRS TO BE PASSED IN THIS ',
        2'MODE, ALREADY ',F8.5,' A-HRS',/,1X,'HAVE BEEN PASSED.')
        GOTO (25,35) IHOLD+1
C
C               STOP request
C
25      WRITE(NN1,1700) JCEL
1700    FORMAT(1X,'YOU NOW MAKE A REQUEST TO STOP THE EXPERIMENT ON ',
```

```
     1'CELL ',I2,' IN A',/,1X,'SPECIFIED AMOUNT OF TIME. THE CELL ',
     2'WILL STOP CYCLING AND IT''S',/,1X,'PARAMETERS WILL BE SAVED ',
     3'ON A FILE.',/,1X,'NOTE THAT THIS WILL ONLY BE CONSIDERED AS ',
     4'AN INTERRUPTION OF A HALF-CYCLE,',/,1X,'WHICH MEANS THAT THE',
     5' SAME HALF-CYCLE WILL BE COMPLETED ON RESUMPTION',/,1X,'OF ',
     6'THE EXPERIMENT.',/,1X,'IS THIS WHAT YOU WANT ? TYPE Y OR N')
      READ(5,1800) ANS
1800  FORMAT(A1)
C     if STOP request does not go through, jump to abort point
      IF(ANS.NE.'Y') GOTO 30
      MSG(1)='T2'
C     ask for time
      WRITE(NN1,1900)
1900  FORMAT(1X,'GIVE HRS,MINS,SECS,TICKS UNTIL INTERRUPTION OF THE ',
     1'EXPERIMENT')
      READ(5,*) MSG(3),MSG(4),MSG(5),MSG(6)
      WRITE(NN1,2000) (MSG(2+KK),KK=1,4)
2000  FORMAT(1X,'YOU WANT THE EXPERIMENT ON THIS CELL TO STOP IN',/,
     11X,I2,' HRS, ',I2,' MINS, ',I2,' SECS, ',I2,' TICKS.',/,1X,
     2'ARE YOU SURE ? Y OR N')
      READ(5,1800) ANS
```

```
C       if information no good, re-try
        IF(ANS.NE.'Y') GOTO 25
C       send message
        NFG=ISDATW(MSG,6)
C       if no foreground active, jump back
        IF(NFG.EQ.1) GOTO 1
C       wait for response
        II=IRCVDW(RESP,2)
C       if not 'OK' response, communication problem
        IF(RESP(2).EQ.'OK') WRITE(NN1,2100)
2100    FORMAT(1X,'TRANSFER COMPLETED')
        IF(RESP(2).NE.'OK') WRITE(NN1,2200) RESP(2)
2200    FORMAT(1X,'COMMUNICATION PROBLEM',/,1X,'CODE RETURNED FROM FG',
        1' IS ',A2)
        RETURN
C       abort
30      MSG(1)='BD'
C       send abort message
        NFG=ISDATW(MSG,1)
C       if no foreground active, jump back
        IF(NFG.EQ.1) GOTO 1
C       wait for response
        II=IRCVDW(RESP,2)
C       if response is not 'BD', then communication problem
        IF(RESP(2).NE.'BD') WRITE(NN1,2200) RESP(2)
C       if response is 'BD', then properly aborted
        IF(IHOLD.EQ.0) WRITE(NN1,2300)
```

```
2300    FORMAT(1X,'STOP')
        IF(IHOLD.EQ.1) WRITE(NN1,2400)
2400    FORMAT(1X,'HOLD')
        WRITE(NN1,2500)
2500    FORMAT('+',6X,'REQUEST ABORTED')
        RETURN
C
C               HOLD request
C
35      WRITE(NN1,2600) JCEL
2600    FORMAT(1X,'YOU NOW MAKE A REQUEST TO STOP THE EXPERIMENT ON ',
       1'CELL ',I2,' AT THE',/,1X,'END OF "A" HALF-CYCLE TO COME.',
       2' THIS CELL, AND ANY OTHER CELL',/,1X,'ATTACHED TO IT, WILL ',
       3'BE PUT IN OPEN-CIRCUIT.')
        IF(RESP(11).NE.-1) WRITE(NN1,2700)RESP(10),(MODENM(KK,
       1(RESP(11)+1)),KK=1,9)
2700    FORMAT(1X,'THERE ALREADY IS A SIMILAR TYPE OF REQUEST PENDING ',
       1'FOR THIS CELL. ',/,1X,'THE EXPERIMENT IS SUPPOSED TO STOP ',
       2'AFTER HALF-CYCLE ',I3,'-',9A1,/,1X,'IF YOU ABORT THE PRESENT',
       3' REQUEST, THE ONE PENDING WILL REMAIN ACTIVE')
        WRITE(NN1,2800)
```

```
2800    FORMAT(1X,'ARE YOU SURE YOU WANT TO GO THROUGH WITH THIS ? TYPE',
        1' Y OR N')
        READ(5,1800) ANS
C       if HOLD request does not go through, jump back to abort point
        IF(ANS.NE.'Y') GOTO 30
        MSG(1)='T3'
C       enter information
        WRITE(NN1,2900)
2900    FORMAT(1X,'GIVE CYCLE NUMBER,MODE (1 FOR CH, 0 FOR DS) OF THE ',
        1'LAST HALF-CYCLE',/,1X,'BEFORE INTERRUPTION')
        READ(5,*) MSG(3),MSG(4)
        WRITE(NN1,3000) (MODENM(KK,(MSG(4)+1)),KK=1,9),MSG(3)
3000    FORMAT(1X,'YOU WANT THE EXPERIMENT STOPPED AFTER THE ',9A1,
        1' OF CYCLE ',I3,'.',/,1X,'IS THIS OK ? Y OR N')
        READ(5,1800) ANS
C       if information no good, re-try
        IF(ANS.NE.'Y') GOTO 35
C       send message
        NFG=ISDATW(MSG,4)
C       if no foreground active, jump back
        IF(NFG.EQ.1) GOTO 1
C       wait for response
        II=IRCVDW(RESP,2)
C       if 'OK', jump
        IF(RESP(2).EQ.'OK') GOTO 45
C       if 'NG', then properly aborted, jump
        IF(RESP(2).EQ.'NG') GOTO 40
```

```
C       communication problem
        WRITE(NN1,2200) RESP(2)
        RETURN
40      WRITE(NN1,3100)
3100    FORMAT(1X,'REQUEST CANNOT BE CARRIED OUT: THE EXPERIMENT IS ',
       1'ALREADY PAST THE ',/,1X,'POINT WHERE YOU WANT TO STOP IT')
C       re-try request
        GOTO 35
45      WRITE(NN1,2100)
        RETURN
        END
1
```

```
      SUBROUTINE RUN
      INTEGER MSG(7),RESP(14)
      INTEGER*4 JTOM
      LOGICAL*1 MODENM(9,2),ANS,ASTRNG(8)
      DATA MODENM/'D','I','S','C','H','A','R','G','E','C','H','A','R',
     1'G','E',' ',' ',' '/
      EQUIVALENCE(JTOM,RESP(6))
      MSG(7)=0
      NN1=20
      NN2=30
      WRITE(NN1,1000)
1000  FORMAT(1X,'CELL NUMBER ?')
      READ(5,*) JCEL
      IF(JCEL.LT.1.OR.JCEL.GT.16) TYPE *,'INVALID CELL NUMBER'
      IF(JCEL.LT.1.OR.JCEL.GT.16) RETURN
1     MSG(1)='RR'
      MSG(2)=JCEL
C     send message
      NFG=ISDATW(MSG,2)
      IF(NFG.EQ.0) GOTO 5
C     if no FG active, invalid request
2     WRITE(NN1,1100)
1100  FORMAT(1X,'NO FOREGROUND JOB ACTIVE. YOUR REQUEST IS INVALID')
      RETURN
C     wait for response
5     II=IRCVDW(RESP,13)
```

```
C       if 'OK', jump
        IF(RESP(2).EQ.'OK') GOTO 10
C       if not 'NG' or 'TH', communication problem
        IF(RESP(2).NE.'NG'.AND.RESP(2).NE.'TH') WRITE(NN1,1200)
1200    FORMAT(1X,'COMMUNICATION PROBLEM')
C       request aborted
        IF(RESP(2).EQ.'TH') WRITE(NN1,1250) JCEL
1250    FORMAT(1X,'THIS CELL NUMBER (',I2,') IS HIGHER THAN WHAT THE ',
       1'REAL-TIME PROGRAM ',/,1X,'KNOWS ABOUT. IF YOU WANT TO RUN ',
       2'THIS CELL, YOU ARE GOING TO HAVE ',/,1X,'TO STOP THE PROGRAM')
        IF(RESP(2).EQ.'NG') WRITE(NN1,1275) JCEL
1275    FORMAT(1X,'CELL ',I2,' IS ALREADY RUNNING. YOUR REQUEST IS ',/,
       11X,'INVALID')
        WRITE(NN1,1300)
1300    FORMAT(1X,'RUN PROCEDURE ABORTED')
        RETURN
10      WRITE(NN1,1400)
1400    FORMAT(3X,'IF NO OPERATION, TYPE 0',/,3X,'IF INITIALIZE, TYPE 1',
       1/,3X,'IF RE-START, TYPE 2')
        READ(5,*) INIT
        GOTO(20,25,15) INIT+1
```

```
C
C               RUN 're-start' mode
C
C       show how the cell was running
15      CALL JJCVT(JTOM)
        CALL TIMASC(JTOM,ASTRNG)
        WRITE(NN1,1500) JCEL,RESP(3),RESP(4),RESP(5),ASTRNG
1500    FORMAT(1X,'CELL ',I2,' STOPPED OPERATING ON ',I2,'-',I2,'-',I2,
        1', AT ',8A1)
        IF(RESP(8).EQ.1) WRITE(NN1,1600)
1600    FORMAT(1X,'IT DID SO BECAUSE THE PROGRAM CRASHED')
        WRITE(NN1,1700) (MODENM(KK,(RESP(10)+1)),KK=1,9),RESP(12)
1700    FORMAT(1X,'THE LAST ACTIVE MODE FOR THIS CELL WAS ',9A1,
        1', ON CYCLE ',I3,'.')
        IF(RESP(9).EQ.0) WRITE(NN1,1800)
1800    FORMAT(1X,'IT WAS PUT IN OPEN-CIRCUIT BY THE PROGRAM, SO IT ',
        1'IS NOW READY',/,1X,'FOR THE NEXT MODE.')
        WRITE(NN1,1900) RESP(11)
1900    FORMAT(1X,'THE LAST BLOCK ON THE SCRATCH-FILE THAT HAS BEEN ',
        1'WRITTEN TO IS',/,1X,'BLOCK NUMBER ',I3,'.')
        WRITE(NN1,2000) JCEL
2000    FORMAT(1X,'ARE YOU SURE YOU WANT TO RE-START CELL ',I2,
        1' ? TYPE Y OR N')
        READ(5,2100) ANS
2100    FORMAT(A1)
C       if RUN 're-start' request does not go through, jump to
C       abort point
```

```
      IF(ANS.NE.'Y') GOTO 20
C     cycle number is the one transferred
      MSG(4)=RESP(12)
C     if in open-circuit, and next mode is one where the cycle number
C     gets incremented, then increment it
      IF(RESP(9).EQ.0.AND.RESP(10).NE.RESP(13)) MSG(4)=RESP(12)+1
      MSG(5)=RESP(13)
C     jump
      GOTO 30
C     abort
20    MSG(1)='BD'
C     send abort message
      NFG=ISDATW(MSG,1)
C     if no FG active, jump back
      IF(NFG.EQ.1) GOTO 2
C     wait for response
      II=IRCVDW(RESP,2)
C     if not 'BD', communication problem
      IF(RESP(2).NE.'BD') WRITE(NN1,1200)
C     request aborted
      WRITE(NN1,1300)
      RETURN
```

```
C
C               RUN 'initialize' mode
C
25      WRITE(NN1,2200) JCEL
2200    FORMAT(1X,'YOU WANT TO INITIALIZE CELL ',I2,'. THIS MEANS THAT',
        1' YOU EITHER START A NEW',/,1X,'CELL, AT CYCLE 1, OR RE-START',
        2' ONE WHOSE PARAMETERS HAVE BEEN RE-INITIALIZED.',/,1X,'IN ',
        3'EITHER ONE OF THESE TWO CASES, YOU''D BETTER MAKE SURE YOU ',
        4'HAVE',/,1X,'PROPERLY ENTERED THESE PARAMETERS.',/,1X,
        5'ARE YOU SURE YOU WANT TO INITIALIZE THE CELL ? TYPE Y OR N')
        READ(5,2100) ANS
C       if Run 'initialize' request does not go through, jump to
C       abort point
        IF(ANS.NE.'Y') GOTO 20
C       cell will start at cycle 1
        MSG(4)=1
C       default for mode at the beginning of which a cycle number is
C       updated is charge
        MSG(5)=1
30      MSG(3)=INIT
        WRITE(NN1,2300)
2300    FORMAT(1X,'DO YOU WANT TO START A NEW SCRATCH-FILE ?',/,1X,
        1'HERE, YOU HAVE TO BE VERY CAREFUL: IF THIS IS A NEW CELL, ',
        2'IT IS ALWAYS',/,1X,'BETTER TO START A NEW SCRATCH-FILE. IN ',
        3'ANY CASE, IF YOU DO THAT, WHATEVER',/,1X,'YOU HAD IN THE ',
        4'SCRATCH-FILE FOR THIS CELL NUMBER WILL BE DELETED.',/,1X,
        5'NOW, ANSWER Y OR N TO THE ORIGINAL QUESTION.')
```

```
      READ(5,2100) ANS
      IF(ANS.EQ.'Y') MSG(6)=1
      IF(ANS.NE.'Y') MSG(6)=0
      WRITE(NN1,2400) JCEL,MSG(4),(MODENM(KK,(MSG(5)+1)),KK=1,9)
2400  FORMAT(1X,'CELL ',I2,' WILL START AT CYCLE ',I3,'.',/,1X,
     1'FOR THIS CELL, THE CYCLE NUMBER IS UPDATED AT THE BEGINNING',
     2' OF A ',/,1X,9A1,'.')
      WRITE(NN1,2450)
2450  FORMAT('+',T13,'(THE USUAL CASE IS CHARGE)')
      WRITE(NN1,2500)
2500  FORMAT(1X,'IF YOU NEED TO MODIFY THIS, TYPE Y, OTHERWISE N')
      READ(5,2100) ANS
      IF(ANS.EQ.'N') GOTO 35
      WRITE(NN1,2600)
2600  FORMAT(1X,'GIVE CYCLE NUMBER (INTEGER)')
      READ(5,*) MSG(4)
      WRITE(NN1,2700)
2700  FORMAT(1X,'TYPE 1 IF CYCLE NUMBER TO BE UPDATED AT THE ',
     1'BEGINNING OF A ',/,1X,'CHARGE, 0 IF DISCHARGE')
      READ(5,*) MSG(5)
35    WRITE(NN1,2750)
2750  FORMAT(//,1X,'ON YOUR REQUEST:',/)
      IF(MSG(3).EQ.1) WRITE(NN1,2800) MSG(2)
      IF(MSG(3).EQ.2) WRITE(NN1,2900) MSG(2)
```

```
2800    FORMAT(1X,'CELL ',I2,' WILL BE INITIALIZED.')
2900    FORMAT(1X,'CELL ',I2,' WILL BE RE-STARTED.')
        IF(MSG(6).EQ.1) WRITE(NN1,3000)
        IF(MSG(6).EQ.0) WRITE(NN1,3100)
3000    FORMAT(1X,'A NEW SCRATCH-FILE WILL BE STARTED FOR THIS CELL.')
3100    FORMAT(1X,'DATA WILL BE ADDED TO THE ALREADY EXISTING SCRATCH-',
       1'FILE FOR THIS CELL.')
        WRITE(NN1,2400) MSG(2),MSG(4),(MODENM(KK,MSG(5)+1),KK=1,9)
        WRITE(NN1,3200)
3200    FORMAT(1X,'IF YOU AGREE WITH ALL THIS, TYPE Y, OTHERWISE N')
        READ(5,2100) ANS
C       if information is no good, re-try
        IF(ANS.NE.'Y') GOTO 10
C       get ready to run !
40      WRITE(NN1,3250)
3250    FORMAT(1X,'MAKE SURE THAT THE CONTROLLER USED IS IN COMPUTER ',
       1'MODE AND THAT',/,1X,'THE MAGIC ENABLE BUTTON IS PUSHED.',/,
       21X,'WHEN READY, TYPE / AND RETURN')
        READ(5,*) NDUM
        MSG(1)='T4'
C       increment counter for cells in series
        MSG(7)=MSG(7)+1
C       send message
        NFG=ISDATW(MSG,7)
C       if no FG active, jump back
        IF(NFG.EQ.1) GOTO 2
C       wait for response
```

```
      II=IRCVDW(RESP,3)
C     if not 'OK', jump
      IF(RESP(2).NE.'OK') GOTO 45
      WRITE(NN1,3300)
3300  FORMAT(1X,'TRANSFER COMPLETED')
      RETURN
C     if 'SC', jump
45    IF(RESP(2).EQ.'SC') GOTO 50
C     if 'EF', jump
      IF(RESP(2).EQ.'EF') GOTO 60
      WRITE(NN1,1200)
      RETURN
C     this cell is in series with another
50    WRITE(NN1,3400) MSG(2),RESP(3),MSG(2),RESP(3)
```

```
3400    FORMAT(1X,'CELL ',I2,' HAPPENS TO BE ATTACHED TO CELL ',I2,
       1'.',/,1X,'SUPPOSEDLY, THESE TWO CELLS HAVE BEEN CYCLING ',
       2'TOGETHER SINCE THEY WERE',/,1X,'FIRST INITIALIZED, SO THAT ',
       3'THE CONDITIONS YOU ENTERED FOR CELL ',I2,' ARE',/,1X,'ALSO ',
       4'VALID FOR CELL ',I2,'. HOWEVER, IF THIS IS NOT THE CASE, YOU',
       5' HAVE TO',/,1X,'RE-STATE THESE CONDITIONS. YOU HAVE TO BE ',
       6'EXTREMELY CAREFUL HERE, BECAUSE IF',/,1X,'YOUR TWO CELLS ARE',
       7' NOT WELL MATCHED, YOU WILL END UP HAVING VERY STRANGE',/,1X,
       8'RESULTS!!!',/,1X,'SO, IF YOU WANT THE SAME CONDITIONS, TYPE ',
       9'Y, OTHERWISE N')
       READ(5,2100) ANS
C      if same conditions, jump
       IF(ANS.NE.'N') GOTO 55
C      if different conditions, jump back
       JCEL=RESP(3)
       GOTO 1
55     MSG(2)=RESP(3)
C      jump back, get ready to run !
       GOTO 40
       RETURN
C      request aborted
60     WRITE(NN1,*) 'COULD NOT OPEN FMAT.UNF'
       WRITE(NN1,1300)
       RETURN
       END
```

REFERENCES

1. RT-11 Documentation Directory, Version 4, Vol. 4, Fortran Language Reference Manual, Jan., 1980. Digital Equipment Corporation, Maynard, Massachusetts.

APPENDIX F

THE DATA REDUCING PROGRAMS

F.1 DATA GATHERING ORGANIZATION

The data reducing programs are written in FORTRAN IV programming language for LSI-11/23 computers (1). As explained in Section D.3, the REAL-TIME program takes data for each cell under test, and puts the data in a scratch-data-file. The name of the file is encoded with the number of the cell (DATØJØ.UNF with J from 1 to 9, or DATØJJØ.UNF with JJ from 10 to 16). When a cell is put out of operation (STOP or HOLD request, see sections E.2.5 and E.2.6), its scratch-data-file is closed. The user can copy the file to a backup file, on another disk; the backup file has a name which contains the cell's identification (e.g., P1DT1.UNF for cell PWR1).

When DATØJØ.UNF is full (maximum 256 blocks), the user must ask for a new scratch-file: the REAL-TIME program starts accumulating data back in block Ø of DATØJØ.UNF. A second backup file must be used (e.g. P1DT2.UNF for PWR1) into which the new data are copied. In general, backup files are necessary because if by error, the user asks for a new scratch-file, part of the original data is erased.

The contents of a (backup) scratch-file can be displayed with a program called DATCHK. A (backup) scratch-file can be decoded and put in a final archive format, by an archiving program called

MASSAG; the archive file is given a name that contains the cell's identification (e.g., DTP1.AR1 for PWR1).

Programs that display the contents of the archive file or plot the data are not described in this work (2).

F.2 THE SCRATCH-DATA-FILE DISPLAY PROGRAM : DATCHK

This program has two functions:

1. Search for a cycle in the scratch-file: the user can start the search at the beginning of the file, or at any block in the middle of the file.

2. Display a block of the scratch-file on the CRT or the line-printer.

DATCHK is a virtual job, which runs in the background with the command R DATCHK.

F.3 THE ARCHIVING PROGRAM : MASSAG

The main function of this program is to put the data in an archive file. As other options, the program displays the file header (Section F.3.1.1) and cycle table (Section F.3.1.2), initializes or deactivates the file (Section F.3.1.1), or modifies a cycle leader (Section F.3.1.3). MASSAG is a virtual job, which runs in the background.

F.3.1. STRUCTURE OF THE ARCHIVE FILES

The first block of an archive file is a file header, which contains general information about the file and the cell to which it belongs. The second and third blocks contain a table of cycles and their location in the file. The fourth block is where the data-storage starts. Every cycle has its own header, which is a 40 real-word long area, that is partly filled with information about the particular cycle.

F.3.1.1 THE FILE HEADER

The file header contains the following information:

1) A file-initialize flag (INTEGER). The value of this flag is set to 10 when the file is initialized.

2) The date at which the file was initialized (3 INTEGER).

3) The cell identification (maximum 10 ASCII).

4) The name of the archive file, if any, that preceeds this one and contains data for the same cell (maximum 10 ASCII).

5) The name of the archive file, if any, that follows this one and contains data for the same cell (maximum 10 ASCII).

6) The name of the file which contains the program that can read this active file (maximum 10 ASCII).

7) The total number of cycles stored (INTEGER).

8) The block number at which the cycle table starts (INTEGER).

9) The block number at which the cycle data start (INTEGER).

The file header is in block Ø; the cycle table starts in block 1, and by default, the cycle data start in block 3. Only 27 of the 256 words in block Ø are used, which leaves ample room for expansion.

The program has an initialize function, which permits the user to enter information in the file header. Once the file is initialized, the program will not allow re-initialization, unless the file is first deactivated. This is another function of the program, which erases block Ø of the file.

F.3.1.2 THE CYCLE TABLE

The table of cycles contains a 3-integer-word entry for each cycle stored in the file.

1st word : cycle number

2nd word : number of the block in which data storage for the cycle starts, as an offset, in number of blocks, relative to the last block of the table (2), e.g., if cycle 2 starts in block 5, the offset is 5-2=3.

3rd word : location in the block at which the cycle data start, given in number of integer words (total number of integer words in block is 256).

F.3.1.3 CYCLE HEADER

A cycle header contains the following information:

1) The cycle number (INTEGER).

2) The total number of words for the cycle's data (INTEGER).

3) The data-point at which charge information starts, counting REAL numbers from after the end of the cycle header (INTEGER) (default is 1).

4) Same, for discharge (INTEGER).

5) The number of types of data for each data-record, including time (INTEGER).

6) A string of symbols identifying the types of data (12 ASCII).

7) The A-hrs on charge and discharge (2 REAL).

8) The shunt identification (4 ASCII).

9) The shunt resistance (REAL).

10) The W-hrs on charge and discharge (2 REAL).

11) The average voltage on charge and discharge (2 REAL).

There are 14 words of header space that are unused at this moment.

F.3.1.4 DATA STRUCTURE

The first part of the cycle data are charge data; the second part are discharge data. Each data-record contains a time and a number of measurements associated with that time. Charge data are recognized by their negative current, discharge data by their positive current, and open-circuit data by their zero current. The times are in hours, relative to the start of charge, discharge or open-circuit. The last data-record of any open-circuit period is a time, in hours, indicating how long the open-circuit period lasted.

For example:

TIME	MEAS.1	.....	MEAS.X	CURRENT	
0.	.....		.....	-.5	charge
.	.....		.....	.	
.	.....		.....	.	
.	.....		.....	.	
2.	.....		.....	-.5	
0.	.....		.....	0.	open-circuit
.5	.....		.....	0.	
6.	0.		0.	0.	Length of open circuit is 6 hours
2.	.....		.....	-.5	continuation of charge
.	.....		.....	.	
.	.....		.....	.	
.	.....		.....	.	
5.	.....		.....	-.5	
0.	.....		.....	0.	open-circuit
.5	.....		.....	0.	
1.	.....		.....	0.	
1.5	0.		0.	0.	length of open circuit is 1.5 hr
0.	.....		.....	.9	discharge
.	.....		.....	.	
.	.....		.....	.	
3.	.....		.....	.9	
0.	.....		.....	0.	open-circuit
.5	.....		.....	0.	
1.	.....		.....	0.	
1.5	0.		0.	0.	length of open circuit is 1.5 hrs

F.3.2 TRANSFER OF DATA FROM A SCRATCH-FILE TO AN ARCHIVE FILE

The archiving program is invoked with the command R MASSAG. The user must specify the following information:

1) The name of the archive file where the data will get stored

2) The name of the scratch-file from which the data is retrieved

3) Whether the transfer starts at the beginning or the middle of the scratch-file

4) The number of the first cycle to be transferred

5) The number of cycles to be transferred

6) If the transfer starts in the middle of the scratch-file, then the following information must be specified for a search:

   6a) whether the search starts in the beginning of the scratch-file or not

   6b) if 6a) answered NO, then the block number at which it starts

   6c) the number of measurements per data-record (time not included) for the length of the file over which the program must search

   The following information is on a cycle-per-cycle basis: for each point, the user must specify for how many cycles, among those to be transferred, the information is valid.

7) The number of data types per data-record and their identification.

8) Whether default storage (= all data-records) is selected or not; if not default then some parameters for data-screening.

9) Whether or not the potential of each cell electrode was measured against the reference electrode; if yes, the discrepancy allowed between the sum of these two measurements and the cell voltage.

10) The measuring shunt identification and its resistance in Ohms.

F.3.3 NOTES

1) The program does not correctly handle cycle sequences with missing cycles

2) The maximum number of data-records per cycle is 150. If the actual number exceeds 150, non-default storage must be used.

3) The program can fail if the scratch-data file is corrupted (see next section) or if the user tries to transfer more cycles than the scratch-file contains. In that case, the user should re-initialize the archive file and completely re-start the archiving process for that cell.

F.4 PATCHING A SCRATCH-FILE - DATPAT

Each block of a scratch-data-file is read according to the values of the pointers stored in the block's pointer area (see D.3). If a block is partly corrupted, the user can change the way it is read by modifying the pointers. This typically happens after a crash of the REAL-TIME

program. The NSTOP flag (see D.3) for the last block that was written into must be set to a value different from zero, equal to the number of valid data-records in the block. The user must run program DATPAT with the command R DATPAT, which allows him/her to change the pointers and the A-hrs of any block in a scratch-file.

Data for PWR-cells can be found:

1) on disk 8, in the scratch-files with name PJDT1.UNF and/or PJDT2.UNF, with J:1→7;
2) on disk 11, in the archive-files with name DTPJ.ARI, with J:1→7.

REFERENCES

1) RT-11 Documentation Directory, Version 4, Vol. 4, Fortran Language Reference Manual, January 1980. Digital Equipment Corporation, Maynard, Massachusetts.

2) Jim Nichols, M.S. Thesis, University of California at Berkeley (1982).

APPENDIX G

APPLICATION OF THE EPA DRIVING PROFILE TO Zn/NiOOH CELL TESTING

G.1 INTRODUCTION

The EPA Urban Driving profile is shown in Figure 1. It plots the speed of a vehicle in urban driving conditions over a total time of 1372 seconds, and an approximate distance of 7.5 miles.

G.1.1 BATTERY POWER

The driving profile can be converted to an equivalent power profile, by using the velocity of the vehicle and the resistive forces to which it is subjected. These include acceleration (linear and rotational) and aerodynamic drag force, rolling resistance of the tires and the force to climb a grade. The resulting expression for the power needed to propel the vehicle is given by Cairns et.al. (1):

$$P_p = M_v V \left[1.1\, \Delta V/\Delta t + gK(1*4.7\times10^{-3}V + 1.3\times10^{-4}\, V^2) + g \sin\theta\right] + \rho_A C_D A_f\, V^3/2$$

with

P_p = propulsion power at the wheels (Watts)

M_v = mass of the vehicle (kg)

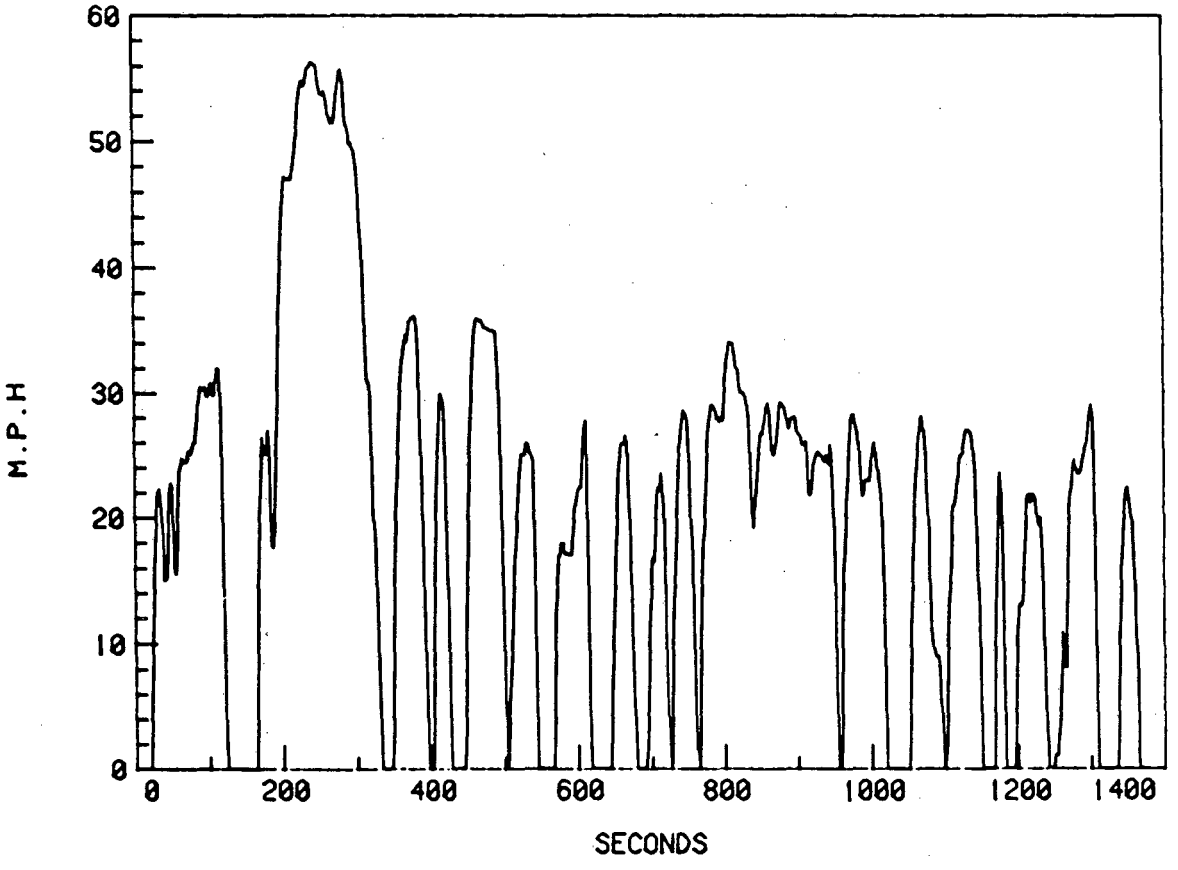
V = velocity of the vehicle (m/s)

t = time (s)

g = gravitational constant (9.80665 m/s^2

K = rolling resistance constant (dimensionless)

θ = angle of the grade



EPA URBAN DRIVING PROFILE : VELOCITIES

XBL 826-801

FIGURE 1. EPA Urban Driving Profile, Federal Register 11/10/70.

ρ = density of air

(1.2255 kg/m^3 at 15°C and 1 atm)

C_D = air drag coefficient (dimensionless)

A_f = frontal area of the vehicle (m^3)

The power to be supplied by the battery is

$$P_b = P_p/\eta_m\eta_e + P_{acc}$$

with

η_m = mechanical efficiency of the drive train

η_e = electrical efficiency of the drive train

P_{acc} = accessory power required from the battery

The values selected for this application were:

M_v = 1000 kg

K = .012

C_D = .45

θ = 0.

A_f = 2 m^2

$\eta_m \eta_e$ = .7

P_{acc} = 250W (no air conditioning)

G.1.2 SCALING OF POWER REQUIREMENTS FOR A Zn/NiOOH CELL

The power to be delivered by a cell was calculated as

$$P_{cell} = P_b \times M_{cell}/M_b$$

With M_{cell} the weight of a cell and M_b the weight of the battery. The weight of the battery was taken as 300 kg, for a 30% battery fraction (M_b/M_v).

The weight of a 2.6 A-hr Zn/NiOOH cell was estimated as follows: average discharge voltage = 1.6 V → energy delivered on discharge is

$$2.6 \text{ A-hr} \times 1.6 \text{ V} = 4.16 \text{ Wh}$$

with a state-of-the-art specific energy of 70 Wh/kg, one gets:

$$4.16 \text{ Wh}/70 \text{ Wh/kg} = .059 \text{ kg}$$

The discharge power profile is shown in Figure 2.

Four computer programs were written for this application: they are described in the following sections. Note that none of these programs have been written to work with the Extended Memory Monitor.

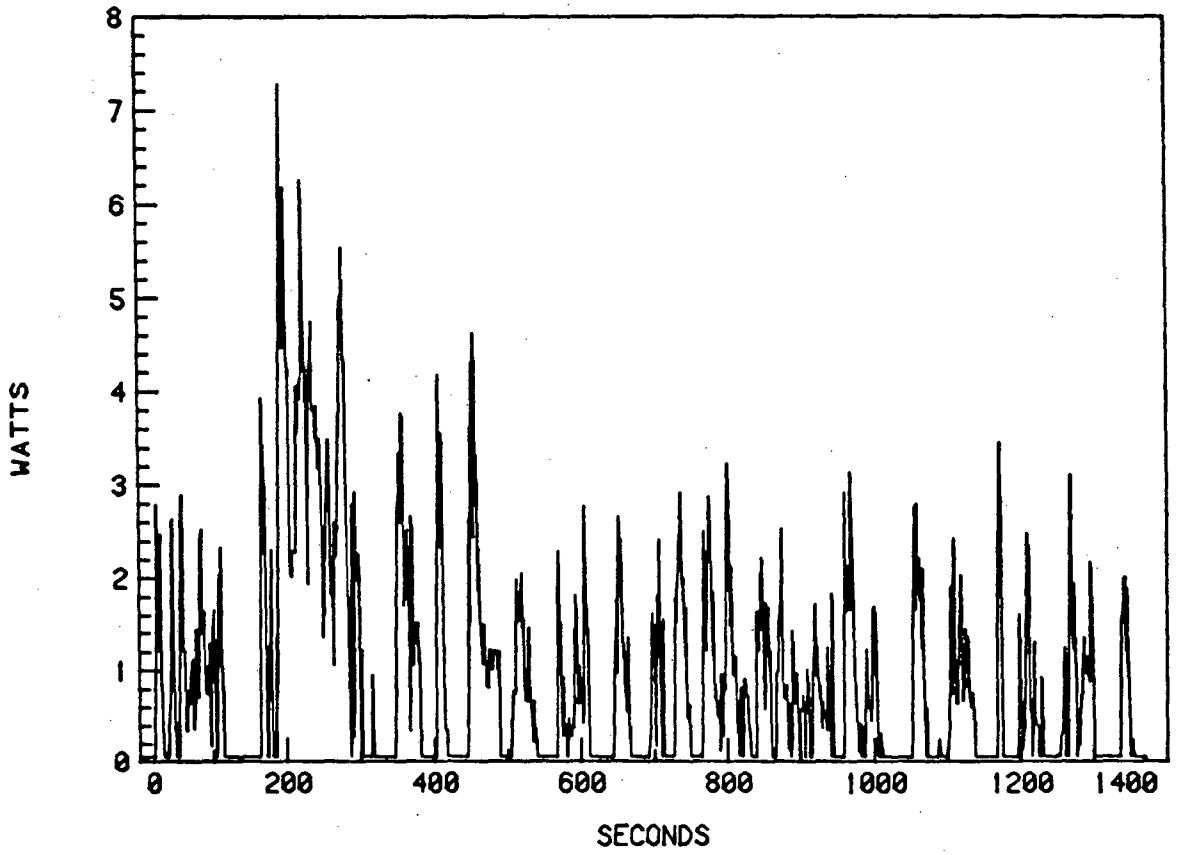


FIGURE 2. EPA Urban Driving Profile, Federal Register 11/10/70.

Power requirements are scaled to a 2.6 A-hr Zn/NiOOH cell.

G.2 THE POWERS PROGRAM

This program has five functions:

1) Initialize the files containing the velocity and power tables (VELOC.UNF and POWERS.UNF).

2) Construct the velocity table

3) Construct the power table (from the velocity table)

4) Display the velocity and power tables

5) Perform range and W-hr calculations. The user specifies either a number of kilometers, or a number of W-hrs or a number of profiles, and the two equivalent values are computed.

G.3 THE PLTPWR PROGRAM

This program plots the velocity or power profile.

G.4 THE PRFPLS PROGRAM

This program organizes a cycling test on one cell, on current controller 8. The charge and discharge requirements are scaled for a 2.6 A-hr Zn/NiOOH cell.

1) The cell is charged with a 10ms-on/90ms-off pulse for 5 hours (.52 A average, 5.2 A peak).

2) The cell is repeatedly discharged with the EPA power profile, until its voltage reaches 1 V.

3) The power regulation algorithm is

$$I_{i+1}/I_i = (P_{i+1}/E_{i+1})/(P_i/E_i)$$

With

$$I_{i+1} = I_i + \Delta I$$

and

$$E_{i+1} = E_i + (\Delta E/\Delta I)\,\Delta I.$$

This gives

$$\Delta I = (P_{i+1} - P_i)/(E_i + X_i I_i),$$

with

$$X_i = (E_i - E_{i-1})/(I_i - I_{i-1})$$

(see subroutine CP of the REAL-TIME program).

4) The charge profile is generated by software,using the programmable clock.

5) The program does not use the I/O driver, but rather the regular I/O library.

6) The program uses a file called PRFPLS.UNF for data-storage. This file has a 16 real-word general header, a 7 real-word header for each charge, and a 7 real-word header for each discharge.

7) On charge, the program takes data every 15 minutes, at the on- and off-side of the pulse. On discharge, the program takes data at three power load values:

- base load : .049 W
- peak load : 7.28 W
- intermediate load : 2.90 W

Figure 3 illustrates the power regulation performance of the PRFPLS program, for a 4 A-hr Cd/NiOOH cell.

G.5 THE PRFCHK PROGRAM

This program plots or prints the cycling results for a cell cycled with the PRFPLS program. A typical voltage plot is shown in Figure 4.

REFERENCES

1. E. J. Cairns, and E. H. Hietbrink, "Electrochemical Power for Transportation", from Comprehensive Treatise of Electrochemistry, Vol.3, p. 421, Editors T. O. M. Bockris, B. E. Conway, E. Yeager, and R. E. White, Plenum Publishing Corporation, 1981.

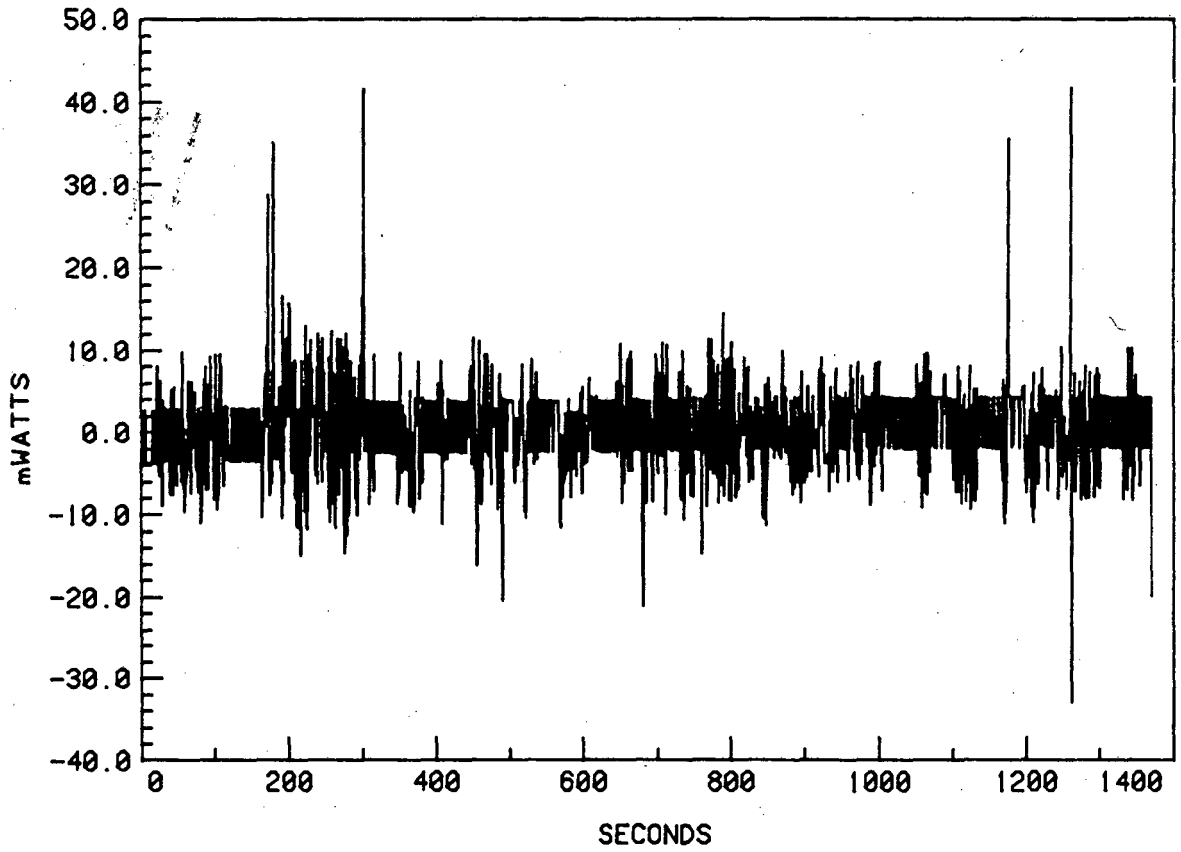
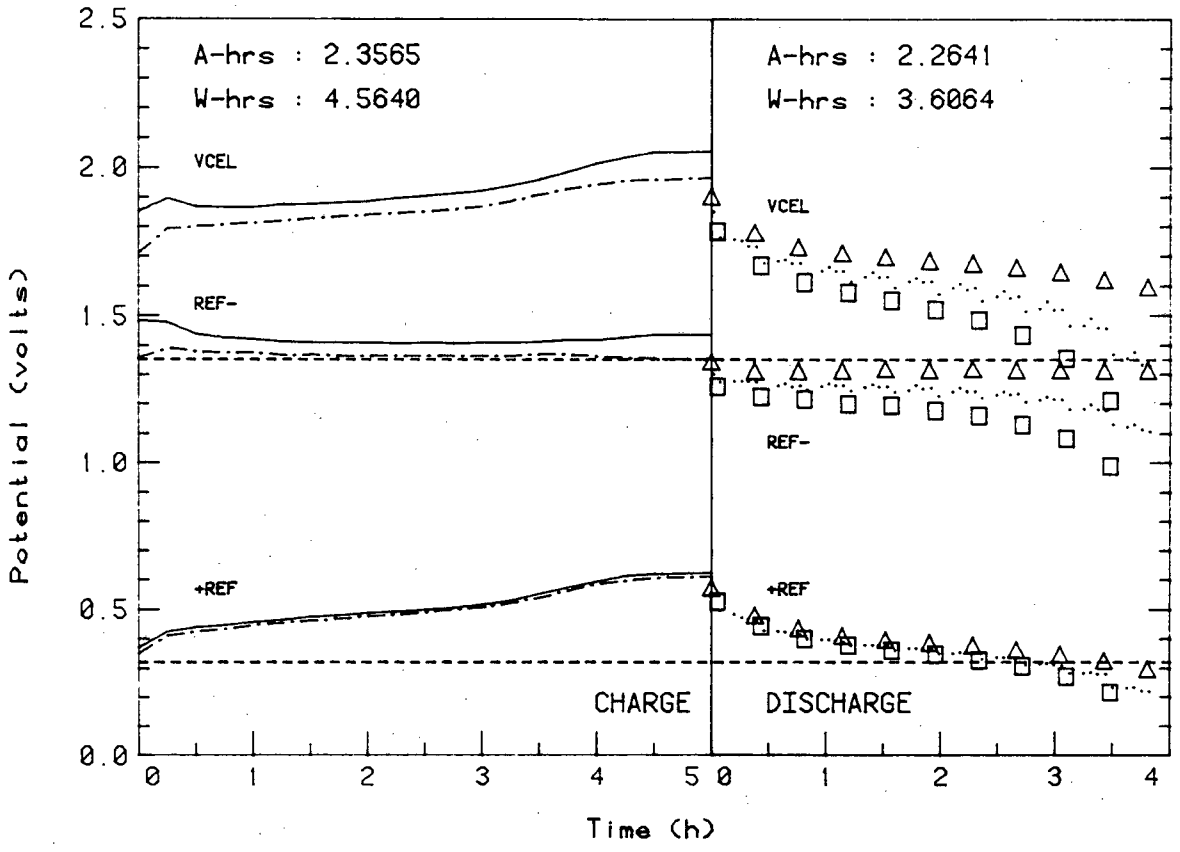


FIGURE 3. Comparison of Computer-Implemented EPA Urban Driving Profile with Specified Profile.

ORDINATE: Specified Power minus Actual Power.

ABSCISSA: Time (sec).



CELL VOLTAGE AND REFERENCE VOLTAGES FOR CYCLE 6 CELL 3A

XBL 826-798

FIGURE 4. Charge/Discharge Curves for a 2.6 Ah Zn/NiOOH Cell Cycled with 10 Hz Pulsed-Current Charging (9/1 off/on) and EPA Power-Profile Discharging.

CHARGE: Solid curves: measurements at current-on time
Dotted-dashed curves: measurements at current-off time
VCEL: NiOOH electrode vs Zn electrode
REF-: Hg/HgO reference electrode vs Zn electrode
+REF: NiOOH electrode vs Hg/HgO reference electrode
Dashed lines: open-circuit values of REF- and +REF

DISCHARGE: Triangles: at base load (0.49 W)
Squares: at peak load (7.28 W)
Dots: at intermediate load (2.90 W)

Other notation as in charge portion of this figure.

This report was done with support from the Department of Energy. Any conclusions or opinions expressed in this report represent solely those of the author(s) and not necessarily those of The Regents of the University of California, the Lawrence Berkeley Laboratory or the Department of Energy.

Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Department of Energy to the exclusion of others that may be suitable.

TECHNICAL INFORMATION DEPARTMENT
LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720